REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre :IVA8/M2/2021

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Image et Vie Artificielle (IVA)**

Une approche pour la détection du texte dans les images

Par :

HADJADJ OUALIE EDDINE

Soutenu le ../../….devant le jury composé de :

Nom Prénom	grade	Président
Nom Prénom	grade	Rapporteur
Nom Prénom	grade	Examinateur

Année universitaire 2020-2021

Dédicace

Je dédie ce modeste travail

À mes très chers parents pour leur soutient et encouragement durant toutes mes années d'études et sans lesquels je n'aurais jamais réussi, À mes frères Fateh et Fouad. À tous mes amis Noufel, Moncif, Rami, Amjed, Wassim et Imad, et à Rayene qui m'ont aidés, soutenus et encouragés. À tous mes enseignants durant mes années d'études avec lesquels j'ai beaucoup appris et mon enseignants et encadrant Dr Tibermacine Ahmed.

Remerciements

Avant tout, mes louanges profondes et sincères à Allah le tout puissant pour m'avoir fourni la force et la patience pour accomplir ce travail de recherche. Je tiens à exprimer toute ma reconnaissance à mon directeur de mémoire, Dr Tibarmacine Ahmed. Je le remercie de m'avoir encadré, orienté, aidé et conseillé.

J'adresse mes sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de me rencontrer et de répondre à mes questions durant mes recherches.

Je remercie mes très chers parents, qui ont toujours été là pour moi. Je remercie mes frère Fateh et Fouad, pour leurs encouragements.

Enfin, je remercie mes amis Noufel, Moncif, Rami, Amjed, Wassim et Imad qui ont toujours été là pour moi. Leur soutien inconditionnel et leurs encouragements ont été d'une grande aide.

À tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.

Résumé

La détection du texte précise et efficace dans des scènes naturelles est une tâche fondamentale mais difficile en vision par ordinateur, en particulier lorsqu'il s'agit de textes orientés arbitrairement. La plupart des méthodes de détection de texte contemporaines sont conçues pour identifier du texte horizontal, qui ne peut pas satisfaire les exigences pratiques de détection pour diverses images du monde réel telles que des flux d'images ou des vidéos. Pour combler cette lacune, nous proposons détecteur de texte en une étape, basé sur l'architecture YOLO, un modèle de réseau neuronal convolutif (CNN) en temps réel robuste pour détecter des textes orientés dans des scènes d'images naturelles. Des expériences de comparaison de référence sont menées sur une base de données populaire, à savoir ICDAR2015, Les résultats indiquent que la méthode proposée surpasse considérablement les méthodes de pointe en termes d'efficacité de détection tout en maintenant une haute précision ; par exemple, la méthode proposée atteint une F-measure de 86% avec une résolution de 720 x 1280 p sur la base de données ICDAR2015.

***Mots-clés* :** Détection du texte, Réseau neuronal convolutif (CNN), Vision par ordinateur, Scènes naturelles, YOLO.

Abstract

Accurate and efficient text detection in natural scenes is a fundamental but difficult task in computer vision, especially when dealing with arbitrarily oriented text. Most contemporary text detection methods are designed to identify horizontal text, which may not meet the practical detection requirements for various real-world images such as image streams or videos. To fill this gap, we propose one stage text detector, based on the YOLO (You Only Look Once) architecture, a robust real-time convolutional neural network (CNN) model for detecting oriented texts in natural image scenes. Benchmark comparison experiments are conducted on a popular database, namely ICDAR2015. The results indicate that the proposed method significantly outperforms advanced methods in terms of detection efficiency while maintaining high precision; for example, the proposed method achieves an F-measure of 86 % with a resolution of 720 x 1280 p based on the ICDAR2015 datasets.

Key-words : Text detection, Convolutional neural network (CNN), Computer vision, Natural scenes, YOLO.

Table des matières

Chapitre 01 : Vision par ordinateur

Chapitre 02 : Apprentissage Machines

Chapitre 03 : Conception et implémentation

Liste des figues

Introduction Générale

Le texte est un outil essentiel de communication et joue un rôle important dans notre vie quotidienne. Il peut être intégré dans des documents ou des scènes comme moyen de transmettre d'informations. L'extraction du texte peut être considérée comme un élément de base principal pour une variété d'applications basées sur la vision par ordinateur, telles que la robotique, l'automatisation industrielle, la recherche d'images, la traduction instantanée, l'assistance automobile et l'analyse de vidéos sportives.

La détection de texte a suscité une attention considérable au cours des dernières décennies, Les textes dans les scènes naturelles, y compris les panneaux de signalisation routière, les panneaux d'affichage, les panneaux de centres commerciaux, etc. jouent un rôle crucial dans notre vie quotidienne, fournissant des informations essentielles sur la société et notre environnement. En tant que condition préalable à la reconnaissance de texte, la détection de texte en temps réel est essentielle dans le processus d'extraction d'informations textuelles et de compréhension du langage naturel. Pour les applications avec des exigences élevées en temps réel, telles que la traduction de texte en temps réel, l'interprétation de texte pour aider les malvoyants, intelligents conduite et navigation autonome par robot, même un léger retard pourrait avoir des conséquences catastrophiques. Par rapport au texte standard sur les documents ou sur Internet, les textes des scènes naturelles sont différents, avec des tailles, des types de polices, des couleurs, des langues et des orientations variés. De plus, ils ont souvent des intensités d'éclairage variables, des arrière-plans complexes et de multiples angles de prise de vue, ce qui pose des problèmes de détection et de reconnaissance de texte.

En particulier lorsqu'il s'agit de textes orientés arbitrairement. La plupart des méthodes de détection de texte contemporaines sont conçues pour identifier du texte horizontal, qui ne peut pas satisfaire les exigences de détection pratiques pour diverses images du monde réel telles que des flux d'images ou des vidéos. Les chercheurs ont fait des percées dans ce domaine avec le développement rapide d'algorithmes d'apprentissage profond et les applications ont connu une grande réussite dans ce domaine. Jusqu'à présent, ils utilisent des méthodes basées sur les réseaux de neurones. Les réseaux de neurones convolutifs (en anglais: *Convolution Neural Netwrok CNN*) ont connu un grand succès dans diverses applications de détection de texte dans les images de scènes naturelles. Dans notre projet, nous sommes intéressés à la création d'un système de détection de texte dans les images de scènes naturelles qui utilise des méthodes de traitement d'images basées sur l'apprentissage profond. Pour atteindre notre

objectif, nous proposons un détecteur de texte en une étape, basé sur l'architecture YOLO (You Only Look Once). Un modèle en temps réel robuste pour détecter des textes orientés arbitrairement dans des scènes d'images naturelles.

L'objectif principal de ce travail consiste à concevoir un système à base capable d'extraire du texte enfoui dans les images de scènes naturelles en utilisant les techniques de l'apprentissage profond (Deep learning « DL »).

Le présent mémoire est organisé en trois chapitres dont les thèmes sont donnés ci-dessous : dans un premier temps, nous introduisons le domaine de vision par ordinateur et ses principales caractéristiques et nous parlerons brièvement dans son historique, ses types d'application dans divers domaines, et les diffèrent techniques parmi les plus récentes en traitement d'images. Le deuxième chapitre nous présentons des généralités la notion des réseaux de neurones artificiels, d'apprentissage automatique et de nouveau paradigme Deep Learning « DL ».Dans le troisième chapitre nous présentons la conception générale de notre proposition de détection de texte dans les images de scènes naturelles, les bases de données de référence utilisé pour évaluer notre modèle, les expérimentations réalisées. Nous allons expliquer les différents bibliothèques et paquets utilisé pour atteindre notre objectif et nous analyserons les résultats obtenus.

Chapitre 01 : Vision par ordinateur.

3.3 Introduction

Dans ce premier chapitre, nous définissons le domaine de vision par ordinateur et ses différentes théories, et découverte de techniques de base utilisées pour le traitement d'images numérique. Nous introduisons également dans ce chapitre les travaux connexes à notre problématique de recherche.

3.4 Vision par ordinateur

3.4.1 Définition

La vision par ordinateur est un domaine scientifique interdisciplinaire qui traite de la façon dont les ordinateurs peuvent acquérir une compréhension de haut niveau à partir d'images ou de vidéos numériques. Du point de vue de l'ingénierie, il cherche à comprendre et à automatiser les tâches que le système visuel humain peut effectuer. Les tâches de vision par ordinateur comprennent des procédés pour acquérir, traiter, analyser et « comprendre » des images numériques, et extraire des données afin de produire des informations numériques ou symboliques, par ex. sous forme de décisions [59].

Dans ce contexte, la compréhension signifie la transformation d'images visuelles (l'entrée de la rétine) en descriptions du monde qui ont un sens pour les processus de pensée et peuvent susciter une action appropriée. Cette compréhension de l'image peut être vue comme l'acquisition d'informations symboliques à partir de données d'image à l'aide de modèles construits à l'aide de la géométrie, de la physique, des statistiques et de la théorie de l'apprentissage[59].

La discipline scientifique de la vision par ordinateur s'intéresse à la théorie des systèmes artificiels qui extraient des informations à partir d'images. Les données d'image peuvent prendre de nombreuses formes, telles que des séquences vidéo, des vues de plusieurs caméras, des données multidimensionnelles à partir d'un scanner 3D ou d'un appareil de numérisation médical. La discipline technologique de la vision par ordinateur cherche à appliquer les modèles théoriques développés à la construction de systèmes de vision par ordinateur[59].

3.4.1 Historique

Le développement de la vision par ordinateur a commencé dans les universités pionnières de l'intelligence artificielle à la fin des années 1960. L'objectif était d'imiter le système visuel humain, première étape pour doter les robots d'un comportement intelligent. En 1966, on

croyait que cela pouvait être réalisé grâce à un projet d'été, en attachant une caméra à un ordinateur et en lui faisant « décrire ce qu'il voyait »[59].

Ce qui distinguait la vision par ordinateur du domaine prédominant du traitement d'images numériques à cette époque était le désir d'extraire une structure tridimensionnelle d'images dans le but de parvenir à une compréhension complète de la scène. Des études dans les années 1970 ont formé les premières bases de nombreux algorithmes de vision par ordinateur qui existent aujourd'hui, y compris l'extraction des bords d'images, l'étiquetage des lignes, la modélisation non polyédrique et polyédrique, la représentation d'objets sous forme d'interconnexions de structures plus petites, le flux optique et estimation de mouvement [59].

La décennie suivante a vu des études basées sur une analyse mathématique plus rigoureuse et des aspects quantitatifs de la vision par ordinateur. Ceux-ci incluent le concept d'espace d'échelle, l'inférence de la forme à partir de divers indices tels que l'ombrage, la texture et la mise au point, et les modèles de contour connus sous le nom de serpents. Les chercheurs ont également réalisé que bon nombre de ces concepts mathématiques pouvaient être traités dans le même cadre d'optimisation que la régularisation et les champs aléatoires de Markov [59].

Dans les années 1990, certains des thèmes de recherche précédents sont devenus plus actifs que les autres. La recherche sur les reconstructions projectives 3D a permis de mieux comprendre l'étalonnage de caméras. Avec l'avènement des méthodes d'optimisation pour la calibration des caméras, on s'est rendu compte que de nombreuses idées avaient déjà été explorées dans la théorie de l'ajustement des faisceaux dans le domaine de la photogrammétrie. Cela a conduit à des méthodes pour des reconstructions 3D éparses de scènes à partir de plusieurs images. Des progrès ont été réalisés sur le problème de la correspondance stéréo dense et d'autres techniques stéréo à vues multiples. Dans le même temps, des variations de coupe graphique ont été utilisées pour résoudre la segmentation d'image[59].

Cette décennie a également marqué la première fois que des techniques d'apprentissage statistique ont été utilisées dans la pratique pour reconnaître les visages dans les images (voir Eigenface). Vers la fin des années 90, un changement important s'est produit avec l'interaction accrue entre les domaines de l'infographie et de la vision par ordinateur. Cela comprenait le rendu basé sur l'image, l'interpolation de vue, l'assemblage d'images panoramiques et le premier rendu de champ lumineux [59].

Des travaux récents ont vu la résurgence des méthodes basées sur les fonctionnalités, utilisées en conjonction avec des techniques d'apprentissage automatique et des cadres d'optimisation complexes, Les progrès des techniques d'apprentissage en profondeur ont donné une nouvelle vie au domaine de la vision par ordinateur. La précision des algorithmes d'apprentissage en profondeur sur plusieurs ensembles de données de vision par ordinateur de référence pour des tâches allant de la classification, de la segmentation et du flux optique a surpassé les méthodes antérieures [59].

3.5 Vision humaine et vision artificielle

Une des particularités des êtres vivants est de pouvoir acquérir des images via l'œil, comme une information, puis de pouvoir l'interpréter via le cerveau. L'enjeu de la vision artificielle est de permettre à un ordinateur de "voir" c'est-à-dire, comme l'homme, de récupérer l'information par l'intermédiaire d'un dispositif d'acquisition d'image puis d'exploiter.

3.5.1 Vision humaine

Comprendre le contenu d'une image est automatique pour l'humain mais difficile pour la machine. La raison en est la complexité du processus de vision humaine qui effectue un traitement parallèle des données dont une partie seulement est purement visuelle œil (≈ 1 million de cellules sensibles), cerveau (≈ des milliards de neurones). Nommer un objet est un processus d'inférence (de déduction) qui nécessite des connaissances non contenues dans l'image. Le cerveau humain n'est pas une machine précise, Il nous est difficile de mesurer exactement la taille des objets. Cependant, nos approximations nous permettent d'aller plus loin dans la reconnaissance.

Les avantages de l'être humain c'est de faculté d'abstraction les modèles des objets que nous utilisons sont très abstraits, Faculté de déduction face à un nouvel objet, nous pouvons le deviner grâce à sa sémantique, pas seulement l'apparence, Faculté d'apprentissage par exemple un bébé connaît peu le monde à sa naissance, mais il apprend peu à peu à le reconnaître,Immersion dans le milieu : nous ne faisons pas que voir les objets, nous les vivons (vision active) et informations a priori : nous prenons pour acquis plusieurs informations qui simplifie l'interprétation (horizon, structure logique d'un objet, bon sens, ...)

3.5.1.1 Yeux humains

3.5.2 Vision artificielle

Une image peut montrer la vue partielle ou totale d'une scène, contenir beaucoup de détails et montrer des objets de différentes échelles (environnement, galaxies, bacteries, ...). La vision artificielle permet de faire certaines choses que la vision humaine ne permet pas et elle permet de voir l'invisible.

La vision artificielle sait traiter les illusions optiques :

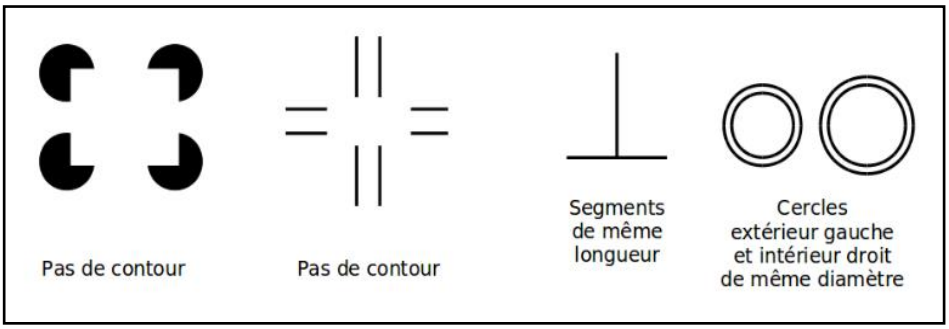

Figure 1.1. Les illusions optiques.

3.6 Application Typique de la vision par ordinateur

3.6.1 Imagerie médicale

La vision par ordinateur aider le médecin lors du diagnostic, le chirurgien lors de la réalisation d'un geste opératoire. Amélioration des images : Rehaussement du contraste, élimination du bruit et mise en évidence des détails. Rehaussement du contraste d'une radiographie de la cage thoracique. Détection des tumeurs cancéreuses à partir d'une mammographie. Recherche de lésions ou d'anomalies dans le cerveau.

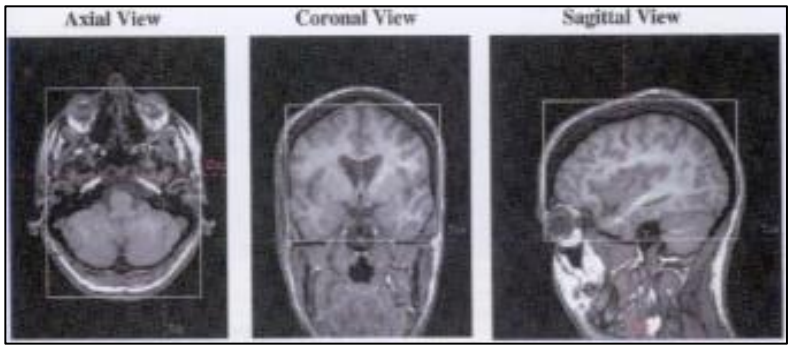

Figure 1.2. Recherche de lésions ou d'anomalies dans le cerveau.

3.6.2 Vision industrielle

Le système de vision détermine la dimension, la forme et la position de l'objet qu'il observe (contrôle dimensionnel). Il détermine la couleur, la texture des objets observés (contrôle d'aspect). A partir des données précédentes, le système détermine la qualité d'un produit (contrôle de la qualité). Il permet ainsi de réduire le nombre d'essais ce qui fait gagner du temps et de l'argent [33] [39].

3.6.3 Extension de perception humaine

Voir ce que l'être humain ne voit pas directement du fait de la limitation de son système visuel. Autres propriétés optiques; Problème d'échelle, de résolution, de point de vue (Images satellitaires). Autres gammes de longueurs d'onde caméras multi-spectrales, infrarouge proche, thermiques (infrarouge lointain). Autres cadences, caméras à haute cadence, plusieurs milliers d'images par seconde.

3.6.4 Energie

Les systèmes de vision sont utilisés pour prévoir la consommation électrique (réduite, normale, élevée), permettant ainsi aux clients de réduire si nécessaire leur consommation, et aux producteurs de mieux gérer leurs unités de production [34] [39].

3.6.5 Lecture automatisée

Les systèmes vision permettent de numériser les anciens documents ainsi que les archives, non pas sous la forme d'images, mais plutôt sous une forme textuelle [35] [39].

3.6.6 Compression d'images

L'objectif de la compression est de réduire la quantité de mémoire nécessaire pour le stockage d'une image ou de réduire le temps de transmission de celle-ci. Compression sans perte : transformation réversible (seul le format des données est modifié, pas le contenu) conserver l'image intacte. Compression avec perte: le contenu de l'image (données numériques) est modifié, mais l'aspect visuel est conservé, autoriser une dégradation de l'image.

Figure 0.3. Image compressée avec JPEG à trois réglages de qualité.

3.7 Un system de vision

Dans les traitements « bas niveau », nous avons besoin de connaissances essentiellement connaissance mathématiques et traitement de l'image. Plus nous montons vers le « haut niveau » (interprétation, compréhension de l'image), plus les connaissances nécessaires changent (intelligence artificielle).

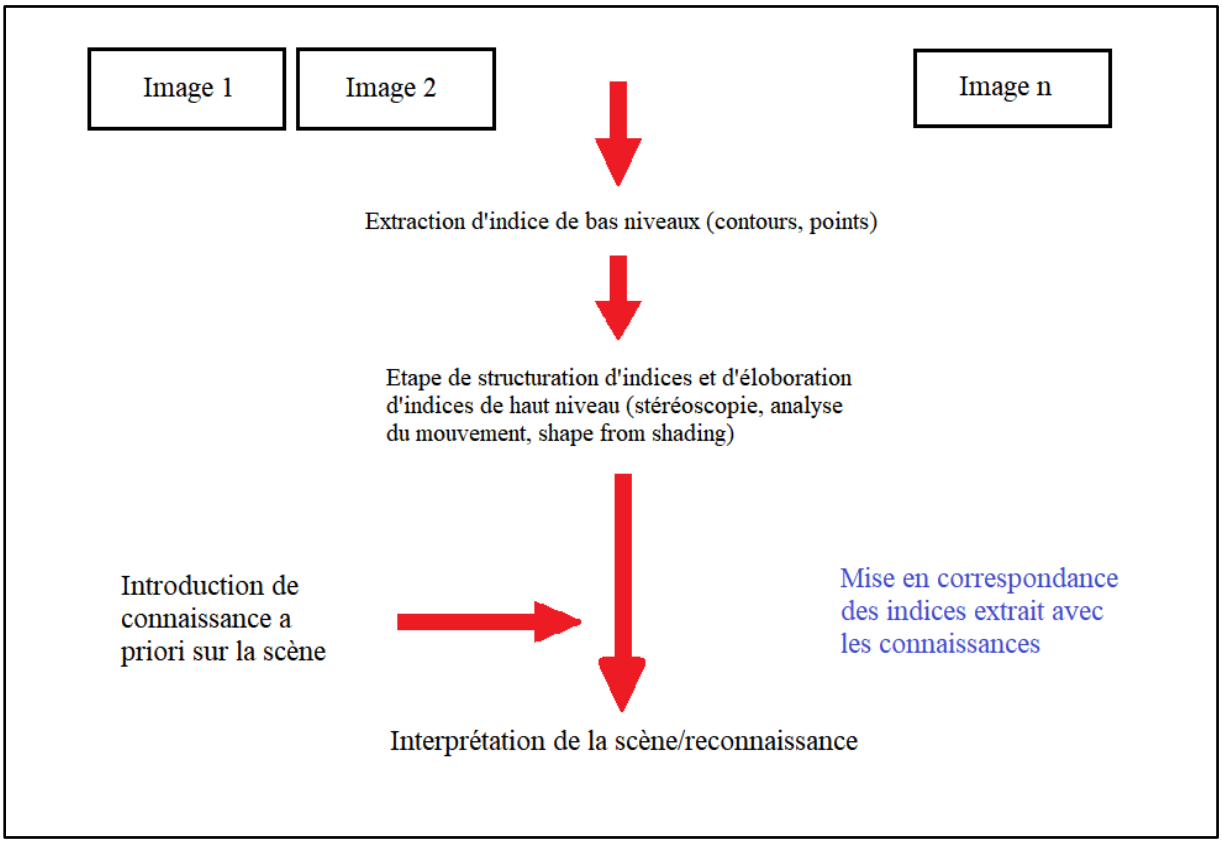

Figure 1.4. Schéma général d'un système de vision.

3.8 Préparation des données

3.8.1.1 Numérisation

À partir des informations du monde physique, construire une représentation des données directement manipulable par la machine. Des capteurs (microphone, caméra, instruments de mesure) convertissent les signaux reçus du monde réel en une représentation numérique

discrète. L'espace résultant, appelé espace de représentation a une dimension **r** très grande lui permettant de disposer du maximum d'informations sur les formes numérisées [39].

3.8.1.2 Prétraitement

Consiste à sélectionner dans l'espace de représentation l'information nécessaire au domaine d'application. Cette sélection passe souvent par l'élimination du bruit, la normalisation des données, ainsi que par la suppression de la redondance. Le nouvel espace de représentation a une dimension **r'** très inférieure à **r** mais demeure un espace de grande dimension et contient des informations encore assez primitives [39].

3.8.1.3 Calcul des représentations

Il s'agit de la phase finale de la préparation des données. Elle fournit un certain nombre de caractéristiques ou paramètres (les fameux attributs) en utilisant des algorithmes de sélection et/ou d'extraction d'attributs. Les attributs étant limités en nombre, l'espace des paramètres ainsi obtenu est de dimension p très petite par rapport à **r'** [39].

3.8.2 Apprentissage

L'apprentissage ou entraînement, est une partie importante du système de vision par ordinateur. Le classificateur étant généralement une fonction paramétrique, l'apprentissage va permettre d'optimiser les paramètres du classificateur pour le problème à résoudre, en utilisant des données d'entraînement. Lorsque les données d'entraînement sont préalablement classées, l'apprentissage est dit supervisé, sinon il est non supervisé [37] [39].

3.8.3 Classification

Cette phase est le noyau de la vision par ordinateur. En utilisant les modèles (paramètres) obtenus lors de l'apprentissage, le classificateur assigne à chaque forme inconnue sa ou ses formes les plus probables [39].

3.8.4 Post traitement

Cette phase a pour but de corriger les résultats de la classification en utilisant des outils spécifiques au domaine d'application. Par exemple pour un système de reconnaissance de textes dans une image, le classificateur se charge de classer chaque caractère séparément, alors que le post traitement appliqué un correcteur orthographique sur tout le texte pour valider et éventuellement corriger le résultat de la classification. Bien que facultative, cette phase permet d'améliorer considérablement la qualité de la reconnaissance [39].

3.9 La vidéo

La vidéo regroupe l'ensemble des techniques, technologie, permettant l'enregistrement ainsi que la restitution d'images animées, accompagnées ou non de son, sur un support adapté à l'électronique et non de type photochimique. Le mot vidéo vient du latin vidéo qui signifie « je vois ». C'est l'apocope de vidéophonie ou vidéogramme.

Un flux vidéo est composé d'une succession d'images, 25 par seconde en Europe (30 par seconde aux USA), composant l'illusion du mouvement. Chaque image est décomposée en lignes horizontales, chaque ligne pouvant être considérée comme une succession de points. La lecture et la restitution d'une image s'effectue donc séquentiellement ligne par ligne comme un texte écrit : de gauche à droite puis de haut en bas.

3.9.1 Caractéristiques de la vidéo

Les principales caractéristiques de la vidéo sont : son nombre d'images par second, son entrelacement, sa résolution d'affichage, son allongement, la couleur espace et de bits par pixel, sa qualité et son début.

Nombre d'images par seconde

Le nombre d'images fixes par unité de temps de la vidéo, le nombre d'images par seconde fut normalisé à 25 images par seconde.

3.10 Traitement d'image

L'image fournie par le capteur est transformée en un signal électrique. De ce signal il faut extraire les informations recherchées sur le contenu de la scène dont l'image a été captée. Les premières bases du traitement d'images sont directement issues du traitement du signal, phénomène normal puisque toute image, qu'elle soit continue ou numérique, peut être considérée comme un signal à 2 dimensions [39].

La vision n'est pas un domaine facile, car repérer un objet simple dans une image demande beaucoup d'opérations. Le traitement, souvent appelé prétraitement, regroupe toutes les techniques visant à améliorer la qualité d'une image. De ce fait, la donnée de départ est l'image initiale et le résultat est également une image [39].

La représentation la plus élémentaire correspond à l'image binaire pour laquelle chaque pixel ne peut prendre qu'une valeur parmi deux autres. Pour les images monochromes, chaque pixel peut prendre une valeur parmi N. N correspond généralement à une puissance de 2, ce qui facilite la représentation de l'image en machine. Par exemple, pour une image en niveau

de gris chacun des pixels peut prendre une valeur parmi 256. Sa valeur est alors codée par un octet de donnée. Une image est constituée par une matrice X lignes et Y colonnes de pixels chacun codé par x bits [39]. Les données quantitatives liées à la représentation des images sont représentées dans le tableau suivant (Tableau2.1) :

Tableau 2. 1. Données quantitatives des images.

1 bit	2 couleurs (noir & blanc)
4 bits	16 couleurs
8 bits	256 couleurs ou niveaux de gris
16 bits	65536 couleurs
24 bits	16 777 216 couleurs (vraies couleurs)
32 bits	4 294 967 296 couleurs

3.10.1 Définition de l'image

Une image est constituée d'un ensemble de points pixels (Picture Element). Il représente le plus petit élément constituant d'une image numérique (on parle d'image numérique lorsque les quantités physiques qui caractérisent l'image sont converties par des valeurs numériques). L'ensemble de ces pixels est contenu dans un tableau à deux dimensions constituant l'image. Les axes de l'image sont orientés de la façon suivante [39] :

- L'axe X est orienté à droite (largeur).
- L'axe Y est orienté de haut en bas (hauteur), contrairement aux notations conventionnelles en mathématiques, ou l'axe Y est orienté vers le haut.

Pour représenter informatiquement une image, il suffit donc de créer un tableau de pixels dont chaque case est codée par un certain nombre de débits déterminant la couleur ou l'intensité du pixel, la figure 2.4 présente un réseau de pixels (L'élément grisé encadré par la couleur rouge correspond aux coordonnées i, j) [39].

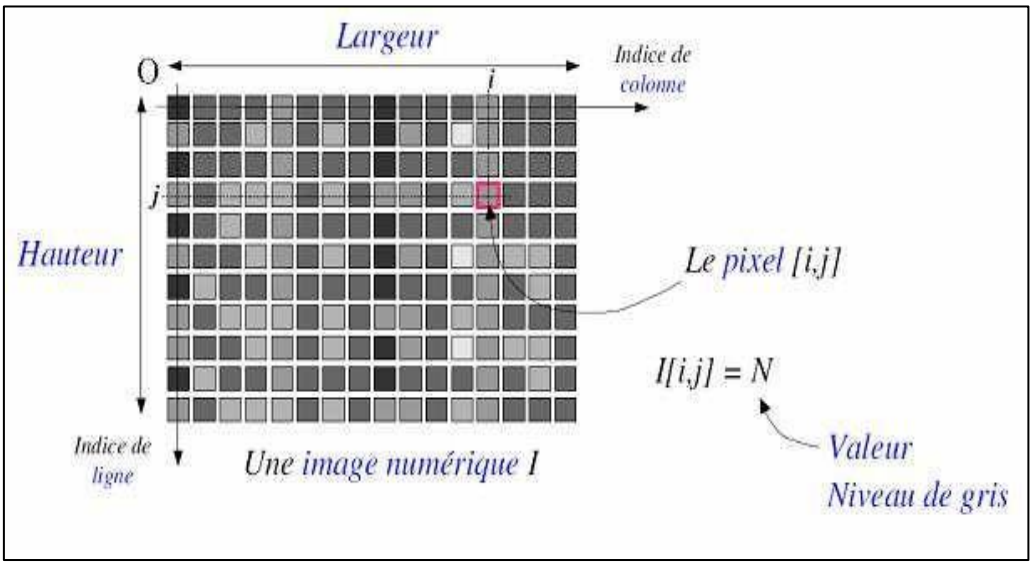

Figure 1.5. Exemple de réseau de pixels. [39].

Pixel : Le terme pixel est la contraction de l'anglais « Picture element». C'est le plus petit élément de l'image. Sortie de l'appareil photo une image est composée d'un nombre x de pixels. Un pixel a une couleur exprimée (codée) en langage binaire mathématique.

3.10.2 Types d'images

3.10.2.1 Image matricielle (bitmap)

Image bitmap est composée d'une matrice (tableau) de points à plusieurs dimensions, chaque dimension représentant une dimension spatiale (hauteur, largeur, profondeur), temporelle (durée) ou autre. Exemple de formats : BMP, PGM, JPEG.

3.10.2.2 Image vectorielle

Une image vectorielle est un ensemble de représentations d'entités géométriques : un cercle, un rectangle ou un segment. Les entités géométriques sont représenté par des formules mathématiques appelées primitives graphiques. Chaque forme élémentaire constitue un objet et se voit assignée un certain nombre d'attributs : la couleur, la transparence, l'épaisseur du trait, le type de trait (pointillé, etc.).

3.10.3 Acquisition d'une image

L'acquisition d'images constitue un des maillons essentiels de toute chaîne de conception et de production d'images. Pour pouvoir manipuler une image sur un système informatique, il est avant tout nécessaire de lui faire subir une transformation qui la rendra lisible et manipulable par ce système. Le passage de cet objet externe (l'image d'origine) à sa représentation interne (dans l'unité de traitement) se fait grâce à une procédure de

numérisation. Ces systèmes de saisie, dénommés optiques, peuvent être classés en deux catégories principales :

- Les caméras numériques.
- Les scanners.

A ce niveau, notons que le principe utilisé par le scanner est de plus en plus adapté aux domaines professionnels utilisant le traitement de l'image comme la télédétection, les arts graphiques, la médecine, etc. Le développement technologique a permis l'apparition de nouveaux périphériques d'acquisition appelés cartes d'acquisition, qui fonctionnent à l'instar des caméras vidéo, grâce à un capteur C.C.D. (Charge Coupled Device). La carte d'acquisition reçoit les images de la caméra, de la T.V. ou du scanner afin de les convertir en informations binaires qui seront stockées dans un fichier [39].

3.10.4 Caractéristiques d'une image numérique

3.10.4.1 Dimension

C'est la taille de l'image. Cette dernière se présente sous forme de matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels). Le nombre de lignes de cette matrice multipliée par le nombre de colonnes nous donne le nombre total de pixels dans une image [39].

3.10.4.2 Résolution

C'est la clarté ou la finesse de détails atteinte par un moniteur ou une imprimante dans la production d'images. Sur les moniteurs d'ordinateurs, La résolution d'une image est définie par un nombre de pixels par unité de longueur de la structure à numériser. Elle se mesure en pixels par pouce (ppp) (*pixel per inch*) points par pouce (1 pouce = 2.54 cm);La résolution d'une image numérique définit le degré de détail de l'image. Plus le nombre de pixels par unité de longueur de la structure à numériser est élevé, plus la quantité d'information qui décrit cette structure est importante et plus la résolution est élevée.[39].

3.10.4.3 Bruit

Un bruit dans une image est considéré comme un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins, il provient de l'éclairage des dispositifs optiques et électroniques du capteur [39].

3.10.4.4 Histogramme

L'histogramme des niveaux de gris ou des couleurs d'une image est une fonction qui donne la fréquence d'apparition de chaque niveau de gris (couleur) dans l'image. Il permet de donner un grand nombre d'information sur la distribution des niveaux de gris (couleur) et de voir entre quelles bornes est repartie la majorité des niveaux de gris (couleur) dans le cas d'une image trop claire ou d'une image trop foncée.

Il peut être utilisé pour améliorer la qualité d'une image (Rehaussement d'image) en introduisant quelques modifications, pour pouvoir extraire les informations utiles de celle-ci. Pour diminuer l'erreur de quantification, pour comparer deux images obtenues sous des éclairages différents, ou encore pour mesurer certaines propriétés sur une image, on modifie souvent l'histogramme correspondant [39].

3.10.4.5 Luminance

C'est le degré de luminosité des points de l'image. Elle est définie aussi comme étant le quotient de l'intensité lumineuse d'une surface par l'aire apparente de cette surface, pour un observateur lointain, le mot luminance est substitué au mot brillance, qui correspond à l'éclat d'un objet. Une bonne luminance se caractérise par :

- Des images lumineuses (brillantes) ;
- Un bon contraste : il faut éviter les images où la gamme de contraste tend vers le blanc ou le noir ; ces images entraînent des pertes de détails dans les zones sombres ou lumineuses.
- L'absence de parasites.

3.10.4.6 Contraste

C'est l'opposition marquée entre deux régions d'une image, plus précisément entre les régions sombres et les régions claires de cette image. Le contraste est défini en fonction des luminances de deux zones d'images. Si L1 et L2 sont les degrés de luminosité respectivement de deux zones voisines A1 et A2 d'une image, le contraste C est défini par le rapport :

$$C = \frac{L_1 + L_2}{L_1 + L_2}$$

3.10.4.7 Images à niveaux de gris

Le niveau de gris est la valeur de l'intensité lumineuse en un point. La couleur du pixel peut prendre des valeurs allant du noir au blanc en passant par un nombre fini de niveaux

intermédiaires. Donc pour représenter les images à niveaux de gris, on peut attribuer à chaque pixel de l'image une valeur correspondant à la quantité de lumière renvoyée. Cette valeur peut être comprise par exemple entre 0 et 255. Chaque pixel n'est donc plus représenté par un bit, mais par un octet. Pour cela, il faut que le matériel utilisé pour afficher l'image soit capable de produire les différents niveaux de gris correspondant. Le nombre de niveaux de gris dépend du nombre de bits utilisés pour décrire la " couleur " de chaque pixel de l'image. Plus ce nombre est important, plus les niveaux possibles sont nombreux.

Il existe plusieurs manières de convertir une image RGB en niveaux de gris. La plus simple est de faire :

$$gris = \frac{rouge + vert + bleu}{3}$$

Cela équivaut aussi à affecter la couleur en niveau de gris à chacune des trois composantes RGB. L'idéal est de faire ressortir la luminosité d'un pixel. Celle-ci vient principalement de la présence de la couleur verte. On emploie généralement les coefficients suivants [39] :

$$\text{Gris} = 0,299 \cdot \text{rouge} + 0,587 \cdot \text{vert} + 0,114 \cdot \text{bleu}$$

Dans ce cas on dispose d'une échelle de teintes de gris, et la plupart du temps on dispose de 256 niveaux de gris avec [68] :

0 ----> noir, ...127 ---> gris moyen, ..., 255 ---> blanc

Certaines images peuvent être codées sur deux octets ou plus (certaines images médicales, des images astronomiques...) ce qui peut poser des problèmes dans la mesure où les systèmes de traitement d'images courants supposent l'utilisation des pixels d'un octet.

3.10.4.8 Images en couleurs

Même s'il est parfois utile de pouvoir représenter des images en noir et blanc, les applications multimédias utilisent le plus souvent des images en couleurs. La représentation des couleurs s'effectue de la même manière que les images monochromes avec cependant quelques particularités. En effet, il faut tout d'abord choisir un modèle de représentation. On peut représenter les couleurs à l'aide de leurs composantes primaires. Les systèmes émettant de la lumière (écrans d'ordinateurs, ...etc.) sont basés sur le principe de la synthèse additive : les couleurs sont composées d'un mélange de rouge, vert et bleu (modèle R.V.B.).

3.10.5 Système de traitement d'images

Un système de traitement numérique d'images est composé de [39]:

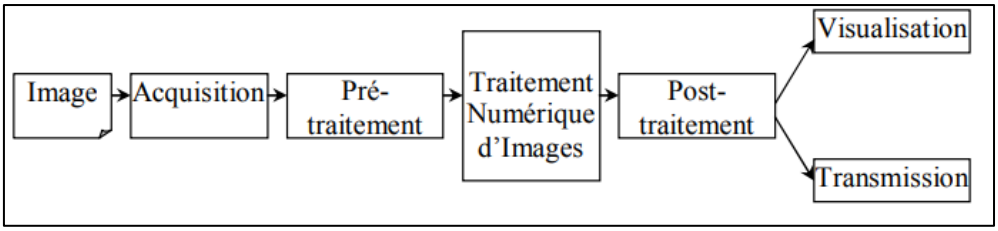

Figure 0.6. Schéma d'un système de traitement d'images.

3.11 Segmentation des images

La segmentation est une étape essentielle en traitement d'image et reste un problème complexe. La segmentation est un processus de la vision par ordinateur, généralement c'est la première étape de l'analyse d'image qui vient après le prétraitement. Le but de la segmentation est d'extraire les informations pertinentes en regard de l'application concernée. La segmentation des images consiste à regrouper les pixels de ces images qui partagent une même propriété pour former des régions connexes. Il existe une multitude de méthodes de segmentation dont l'efficacité reste difficile à évaluer. La segmentation des images est le procédé qui conduit à un découpage de l'image en un nombre fini de régions (ou segments) bien définies qui correspondent à des objets, des parties d'objets ou des groupes d'objets qui apparaissent dans une image. C'est une transformation très utile en vision artificielle [39].

Une erreur dans la segmentation de la forme à reconnaître augmente forcément le risque d'une mauvaise reconnaissance. Essentiellement, l'analyse de l'image fait appel à la segmentation où l'on va tenter d'associer à chaque pixel de l'image un label en s'appuyant sur l'information portée (niveaux de gris ou couleur), sa distribution spatiale sur le support image, des modèles simples (le plus souvent des modèles géométriques) [39].

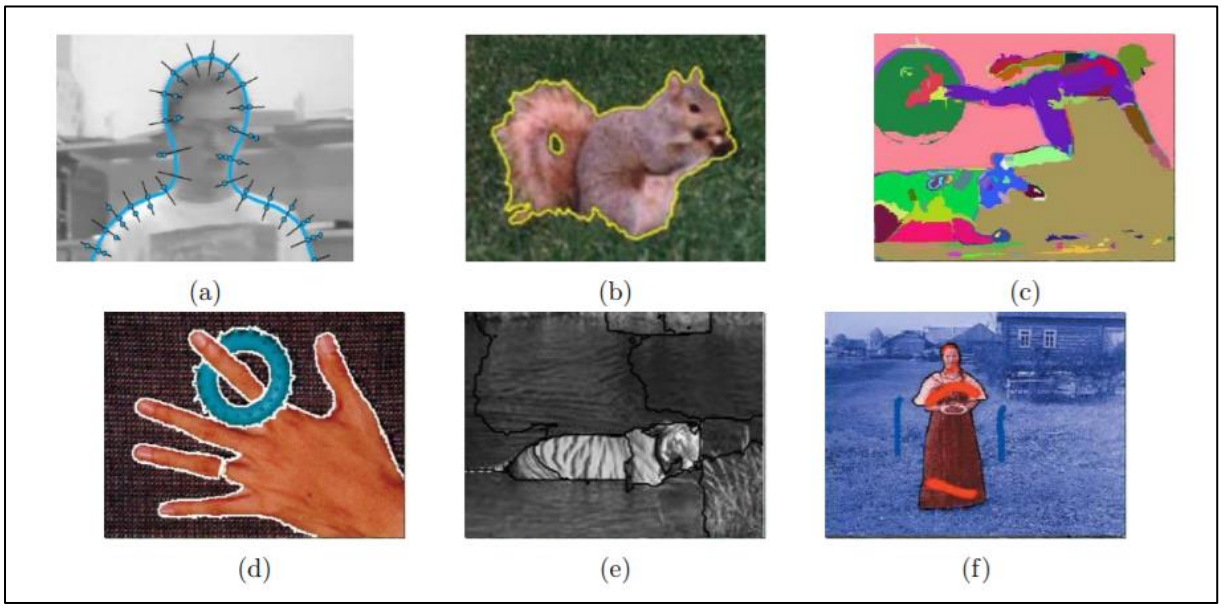

Figure 0.7. Quelques techniques populaires de segmentation d'images : (a) contours actifs(Isard et Blake1998) c 1998 Springer; (b) level sets (Cremers, Rousson et Deriche 2007) c2007 Springer; (c)fusion basée sur des graphes (Felzenszwalb et Huttenlocher 2004b) c 2004 Springer; (d) décalage moyen(Comaniciu et Meer 2002) c 2002 IEEE; (e) la texture et les contours intermédiaires normalisés coupes (Malik, Belongie, Leung et al. 2001) c 2001 Springer ;(f) MRF binaire résolu à l'aide d'un graphique coupes (Boykov et Funka-Lea 2006) c 2006 Springer [1].

3.11.1 Types de Segmentation

3.11.1.1 Segmentation de texte en lignes

En général, le sous-titres ou le texte graphique ce compose de plusieurs lignes. En reconnaissance de texte, le texte se segmentant en lignes séparées. Il existe certaines méthodes utilisées à cet effet, telles que la projection horizontale.

3.11.1.2 Segmentation de lignes en caractères

Il s'agit ici de la segmentation de lignes en caractères individuels. Les points de segmentation sont identifiés à la fin d'un caractère et au début de la suivante.

3.11.2 Les principes de la segmentation

De nombreux travaux ont été réalisés sur ce sujet, dans des domaines aussi variés que le domaine médical, militaire, industriel, géophysique, etc… C'est toujours un sujet d'actualité et un problème qui reste ouvert où l'on retrouve de très nombreuses approches [68] :

- La segmentation par régions ;
- La segmentation par seuillage ;
- La segmentation par contours ;
- La Transformée de Hough ;

- La segmentation par étiquetage en composantes connexes ;
- La segmentation par LPE.

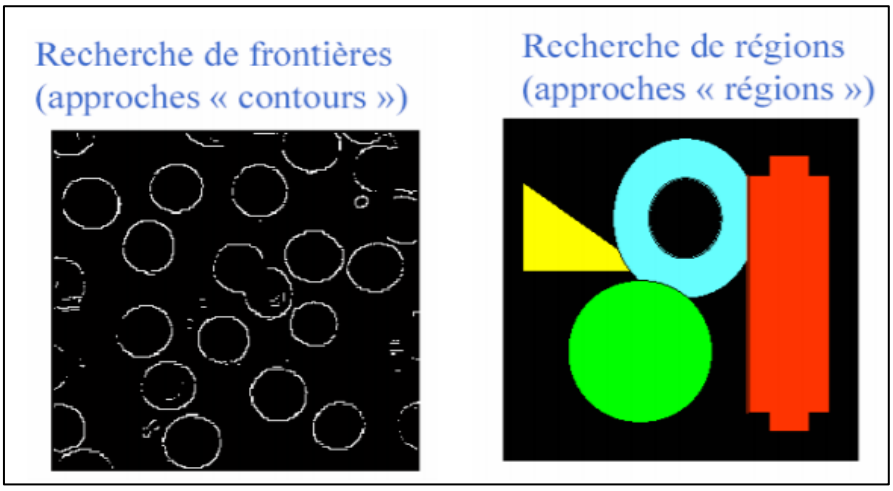

Figure 1.8. Exemples des approches de segmentation.

Toutes ces approches visent à l'extraction des caractéristiques. Après de nombreuses années passées à rechercher la méthode optimale, les chercheurs ont compris que la segmentation idéale n'existait pas. Il n'existe pas d'algorithme universel de segmentation à chaque type d'images correspond une approche spécifique. Une bonne méthode de segmentation sera donc celle qui permettra d'arriver à une bonne interprétation [39].

3.12Travaux connexes

La détection du texte de scène ont été un sujet de recherche actif en vision par ordinateur au cours des dernières décennies. Cette partie est consacrée pour l'analyse des résultats de différents articles qui s'intéressant sur la détection de texte qui utilisent différentes méthodes de l'apprentissage automatique.

3.12.1.1 Le travail de B. Shi, X. Bai et S. Belongie

Dans la recherche de B.Shi, X. Bai et S.Belongie [41]le système réalisé propose Liaison de segments (*SegLink*), une méthode de détection de texte orientée ont utilisé une architecture héritée de SSD. L'idée principale est de décomposer le texte en deux éléments localement détectables, à savoir les segments et les liens. Un segment est une boîte orientée couvrant une partie d'un mot ou d'une ligne de texte ; Un lien relie deux segments adjacents, indiquant qu'ils appartiennent au même mot ou à la même ligne de texte. Les deux éléments sont détectés de manière dense à plusieurs échelles par un réseau de neurones entraîné de bout en bout et entièrement convolutif. Les détections finales sont produites en combinant des segments reliés par des liens. Par rapport aux méthodes précédentes, *SegLink* s'améliore en

termes de précision, de vitesse et de facilité de formation. Il atteint une F-measure de 75,0% sur l'indice de référence standard ICDAR 2015.

3.12.1.2 Le travail de Dorai et al

Dans le travail de Dorai et al [42] proposent une nouvelle méthode de détection de texte dans les images naturelles. La méthode comprend deux contributions : Premièrement, un moteur rapide et évolutif pour générer des images synthétiques de texte encombré. Ce moteur superpose du texte synthétique aux images d'arrière-plan existantes de manière naturelle, en tenant compte de la géométrie de la scène 3D locale. Deuxièmement, nous utilisons les images synthétiques pour former un réseau de régression entièrement convolutif (FCRN) qui effectue efficacement la détection de texte et la régression de la boîte englobante à tous les emplacements et à plusieurs échelles dans une image. Nous discutons de la relation entre le FCRN et le détecteur YOLO récemment introduit, ainsi que d'autres systèmes de détection d'objets de bout en bout basés sur l'apprentissage en profondeur. Le réseau de détection qui en résulte surpasse considérablement les méthodes actuelles de détection de texte dans les images naturelles, atteignant une F-measure de 84,2 % par rapport à la référence standard ICDAR 2013.

3.12.1.3 Le travail de P. Lyu, C. Yao, W. Wu, S. Yan, and X. Bai

Les méthodes de détection de texte de scène de pointe basées sur l'apprentissage en profondeur peuvent être grossièrement classées en deux catégories. La première catégorie traite le texte de la scène comme un type d'objets généraux et suit le paradigme de détection d'objet général pour localiser le texte de la scène en régressant les emplacements des zones de texte, mais troublé par l'orientation arbitraire et les grands rapports d'aspect du texte de la scène. Le second segmente directement les régions de texte, mais nécessite principalement un post-traitement complexe. Dans le travail de P. Lyu, présent une méthode qui combine les idées des deux types de méthodes tout en évitant leurs défauts. Il propose de détecter le texte de la scène en localisant les points d'angle des cadres de délimitation du texte et en segmentant les régions de texte dans des positions relatives. Au stade de l'inférence, les boîtes candidates sont générées par échantillonnage et regroupement de points d'angle, qui sont ensuite notés par des cartes de segmentation et supprimés par NMS. Par rapport aux méthodes précédentes, cette méthode peut gérer naturellement du texte long orienté et ne nécessite pas de post-traitement complexe. Il atteint une F-mesure de 84,3% sur ICDAR2015 et 81,5% sur MSRA-TD500.

3.12.2 Comparaison entres les travaux réalisés

Les travaux cités dans la partie précédents concernant le domaine de la détection de texte dans une image utilisent des différentes bases de données, la comparaison entre eux est faite en fonction des résultats obtenus. Le but est de nous aider à proposer une approche en appliquant des nouvelles techniques sur une base de données plus riches.

Le tableau suivant donne un résumé des résultats des recherches mentionnés :

Tableau 2. 2. Les résultats des travaux connexes.

Articles	La base de donnée	Modèles	Résultats
B. Shi, X. Bai et S. Belongie[41]	ICDAR 2015	Réseaux de neurones convolutionnel	75,0%
Dorai et al [42]	ICDAR 2013	YOLO	84,2 %
P. Lyu, C. Yao, W. Wu, S. Yan, and X. Bai [43]	ICDAR 2015	Réseaux de neurones convolutionnel	84,3%

3.13 Conclusion

La vision par ordinateur est le processus d'identification et de détection d'un objet ou d'une caractéristique d'une image ou d'une vidéo numérique. Ce concept met en œuvre plusieurs étapes : segmentation des objets (analyse d'images), extraction de caractéristiques (géométrie, invariants, …), classification (supervisée ou non, méthodes probabilistes, statistiques, …). Les applications sont très variées et nécessitent des connaissances expertes du domaine d'application. Les méthodes sont nombreuses mais les principes de base sont assez stables. Nous présenterons dans le chapitre suivant le domaine d'apprentissage machines et ses caractéristiques principales et nous avons introduit ses composants.

Chapitre02 : Apprentissage machines.

3.3 Introduction

L'apprentissage automatique (en anglais machine learning, littéralement « l'apprentissage machine ») ou apprentissage statistique est un champ d'étude de l'intelligence artificielle qui se fonde sur des approches statistiques pour donner aux ordinateurs la capacité « apprendre » à partir de données, c'est-à-dire d'améliorer leurs performances à résoudre des tâches sans être explicitement programmés pour chacune. Plus largement, cela concerne la conception, l'analyse, le développement et l'implémentation de telles méthodes.

3.4 Objectif de L'Apprentissage machine

L'apprentissage machine est une sous-discipline de l'intelligence artificiel dont l'objectif ultime est de reproduire chez l'ordinateur les capacités cognitives de l'être humain, par le biais de l'apprentissage. Ici, le terme apprentissage est à mettre en opposition avec des techniques d'intelligence artificielle basées sur des comportements intelligents "pré-codés", comme dans le fameux programme ELIZA [11], où l'ordinateur donnait l'illusion de pouvoir poursuivre une conversation intelligente avec un humain à partir d'un système en fait très basique de détection de mots-clés et de réponses prédéfinie. L'approche "apprentissage machine" consiste plutôt à programmer des mécanismes qui permettent de développer les connaissances c'est-à-dire d'apprendre à partir d'observations (les exemples d'apprentissage) de manière automatique [52].

Les êtres humains faisant preuve de capacités d'apprentissage impressionnantes, il est naturel que l'apprentissage machine s'inspire d'eux pour tenter de reproduire leurs comportements. En particulier, les travaux en neurosciences visant à comprendre les mécanismes d'apprentissage dans le cerveau. Une classe d'algorithmes d'apprentissage très populaire, les réseaux de neurones, a ainsi été inspirée de telles observations biologiques. Mais les détails du fonctionnement du cerveau restant pour l'instant en grande partie un mystère. Les algorithmes développés en apprentissage machine ont des objectifs moins ambitieux. Ils ont chacun leurs propres forces et faiblesses – souvent très différentes de celles des humains – et sont généralement prévus pour résoudre certains types des problèmes spécifique [52].

3.5 Différents types d'apprentissage

3.5.1 Apprentissage supervisé

La forme d'apprentissage qui est considérée comme la plus intuitive est celle dite d'apprentissage supervisé. Cela signifie que la réponse attendue est observée dans les données collectées. Formellement, si l'on note zun exemple d'apprentissage, alors on peut écrire z=(x,y) avec x la partie "entrée",c'est-à-dire les données que l'algorithme est autorisé à utiliser pour faire une prédiction, et yla partie "étiquette", c'est-à-dire la valeur correcte pour la prédiction. Par exemple, si la tâche consiste à déterminer le sexe d'une personne à partir d'une photo d'identité, x serait la photo et ysoit "homme", soit "femme" (en supposant qu'il s'agisse des deux seules possibilités). Dans un tel cas où les valeurs possibles de l'étiquette yne sont pas interprétables comme des nombres, on parle de tâche de classification. Si par contre la tâche était de prédire l'âge de la personne plutôt que son sexe, y serait un nombre et on parlerait d'une tâche de régression. La figure 1.1 illustre ces deux situations [52].

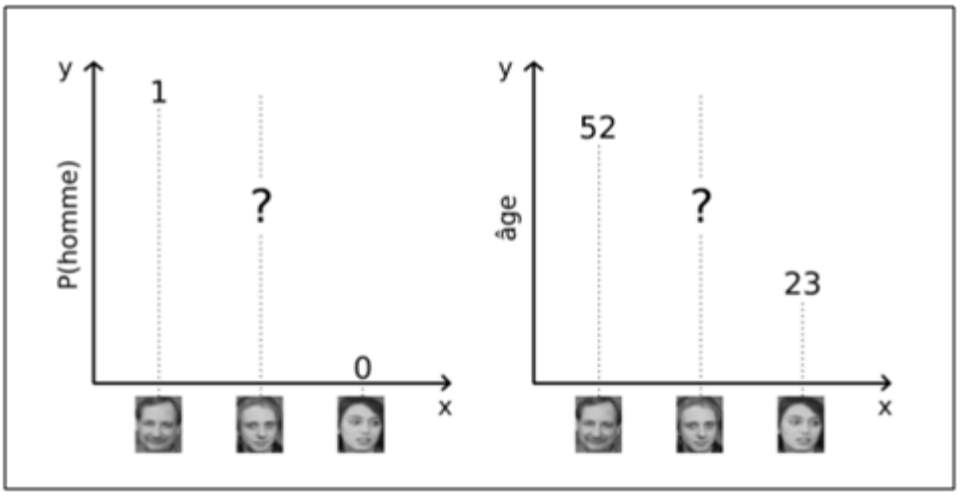

Figure 4.1. Exemples d'apprentissage supervisé. Classification (à gauche) pour prédire le sexe, et régression (à droite) pour prédire l'âge, Les photos aux deux extrémités de l'axe des x sont des exemples d'entraînement pour lesquels l'étiquette est connue, tandis que la photo du milieu est celle pour laquelle on veut obtenir une prédiction [26].

L'algorithme est *entraîné* sur un ensemble de données pré-collectées (l'ensemble d'entrainement), de la forme $D = \{z_1, \dots, z_n\}$, avec $z_i = (x_i, y_i)$. De nombreux algorithmes supposent que ces exemples sont tirés de manière indépendante et identiquement distribuée d'une distribution P, c'est-à-dire. $z_i \sim P(X, Y)$ (où la forme exacte de P n'est pas connue d'avance). Selon la tâche à résoudre, la prédiction d'un algorithme d'apprentissage supervisé va généralement tenter d'approximer l'une des trois quantités suivantes [26] :

– $P(x|y)$: dans notre exemple de classification (trouver le sexe), il faudrait prédire la probabilité qu'une photo soit une photo d'un homme par un nombre p (la probabilité que ce soit la photo d'une femme étant alors 1−p). Dans notre exemple de régression (trouver l'âge), la prédiction serait une densité de probabilité sur l'intervalle $]0, +\infty[$.

– $argmax_y P(x|y)$: dans notre exemple de classification, il s'agirait d'une prédiction binaire du sexe le plus probable ("homme" ou "femme"). Dans notre exemple de régression, il s'agirait de l'âge le plus probable.

– $E_Y = [Y|x] = \int_y yP(y|x)\,dy$: cette quantité n'a de sens que pour la régression, et il s'agirait dans notre exemple de prédire l'âge moyen de la personne étant donnée sa photo.

Il faut noter que la prédiction de $P(y|x)$ est la tâche la plus générale (dans la mesure où la résoudre permet également d'accomplir les deux autres), mais tous les algorithmes d'apprentissage ne sont pas capables d'estimer une distribution de probabilité [26].

3.5.2 Apprentissage non supervisé

Dans l'apprentissage non supervisé, un exemple $z_i = x_i \sim P(x)$ ne contient pas d'étiquette explicite. Il existe plusieurs types de tâches d'apprentissage non supervisé, parmi lesquelles les plus souvent rencontrées sont [26] :

- L'estimation de densité : estimer P(x), comme illustré en figure 2.2.
- La génération de données : tirer de nouveaux exemples d'une distribution la plus proche possible de P(X).
- L'extraction de caractéristiques.
- Le regroupement (clustering) : partitionner les exemples en groupes $G_1, \ldots, G_k$tels que tous les exemples dans un même groupe soient similaires.

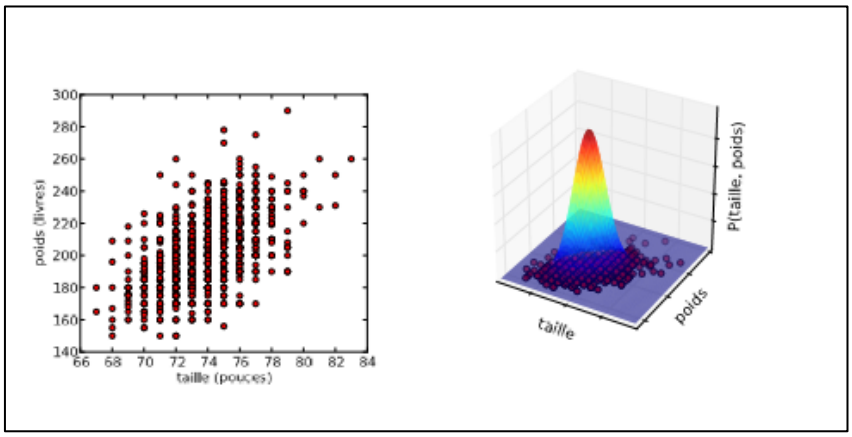

Figure 4.2. Exemple d'apprentissage non supervisé: l'estimation de densité. À gauche, les données d'origine (taille et poids de 1033 joueurs de baseball aux Etats-Unis). À droite, l'estimation de la densité par une distribution Gaussienne [26].

3.5.3 Apprentissage semi-supervisé

Comme le nom l'indique, l'apprentissage semi-supervisé est à mi-chemin entre le supervisé et le non supervisé : certains exemples $z_i = (x_i, y_i)(1 \leq i \leq l)$ ont une étiquette, tandis que d'autres exemples $z_i = x_i(l + 1 \leq i \leq n)$ n'en ont pas (on dit qu'ils ne sont pas "étiquetés"). Les algorithmes d'apprentissage semi-supervisé tentent généralement de résoudre des problèmes d'apprentissage supervisé, mais en utilisant les exemples non étiquetés pour améliorer leur prédiction. La distribution des entrées $P(X)$ peut nous donner de l'information sur $P(Y|X)$ même en l'absence d'étiquettes. Un exemple typique où c'est le cas, pour une tâche de classification, est lorsque les exemples de chaque classe forment des groupes distincts dans l'espace des entrées, séparés par des zones de faible densité $P(x)$. La figure 2.2 montre ainsi deux classes dont la forme en croissant est révélée par les exemples non étiquetés. Un algorithme purement supervisé donc ignorant ces exemples ne pourrait pas identifier correctement ces deux classes [26].

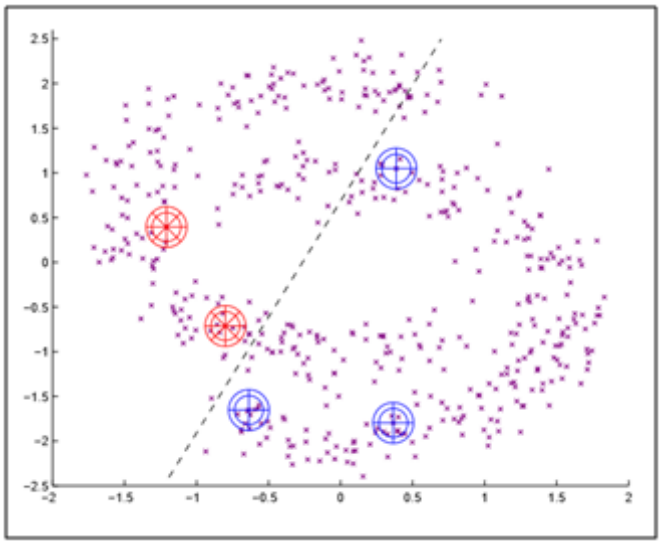

Figure 4.3 Apprentissage semi-supervisé :ici seuls 5 exemples sont étiquetés (deux de la classe * en rouge, et trois de la classe + en bleu). Un algorithme n'utilisant pas les exemples non étiquetés (petits x mauves) séparerait les deux classes par exemple selon la ligne en pointillés, alors qu'un algorithme semi-supervisé (ou un humain) pourrait séparer les deux "croissants de lune" correspondant à chaque classe [26].

Dans le cadre de l'apprentissage supervisé décrit précédemment, l'application d'un algorithme se fait typiquement en deux étapes [26] :

1. L'algorithme va d'abord apprendre une tâche (phase d'entraînement) sur un ensemble $D = \{(x_1, y_1), \cdots, (x_n, y_n)\}$.
2. L'algorithme doit ensuite faire ses prédictions (phase de test) sur un ensemble T dans on ne fournit pas les étiquettes.

Un algorithme semi-supervisé peut également suivre ces deux étapes (en n'oubliant pas que D peut contenir également des exemples non étiquetés). On dit alors que la phase de prédiction sur l'ensemble de test se fait par induction. Mais puisque l'algorithme peut utiliser des exemples non étiquetés au cours de l'apprentissage, on peut également inclure T dans D : on parle alors de *transduction*, et en général les performances seront meilleures qu'en induction puisque plus d'exemples sont disponibles pour l'apprentissage. Il existe malgré tous des applications où il n'est pas possible d'inclure T dans l'ensemble d'entraînement, par exemple lorsque les éléments de T sont générés en temps réel et que l'on a besoin de prédictions immédiates : si ré-entraîner l'algorithme avant chaque nouvelle prédiction s'avère trop lent, l'induction est alors la seule option possible [26].

3.5.4 Apprentissage par renforcement

L'apprentissage par renforcement est une forme d'apprentissage supervisé où l'algorithme n'observe pas une étiquette pour chacune de ses prédictions, mais plutôt une mesure de la qualité de ses prédictions (possiblement prenant en compte toute une séquence de prédictions) [26].

3.6 Généralisation

3.6.1 Sur-apprentissage

L'entraînement d'un algorithme d'apprentissage consiste à extraire, de manière explicite ou implicite, des caractéristiques de la distribution de probabilité P qui génère les données. Mais P étant inconnu, l'algorithme se base à la place sur un nombre fini d'exemples d'entraînement, c'est-à-dire. Sur la distribution discrète $\hat{P}$ des exemples disponibles dans D (appelée la distribution *empirique*). Lorsque certaines caractéristiques apprises sur P ne s'appliquent pas à P, on parle de *sur-apprentissage,* et on risque une mauvaise *généralisation,* c'est-à-dire. Que l'algorithme ne va pas obtenir une bonne performance sur de nouveaux exemples tirés de P. Prenons l'exemple de la classification, lorsqu'un modèle estime $P(y|x)$ par une fonction $q_x(y)$. Afin de mesurer la similarité entre q_x et $P(.|x)$, on peut par exemple utiliser la divergence de Kullback-Leibler D_{KL}[40], définie par [26] :

$$D_{\mathrm{KL}}(P(.|x)||q_x) = \sum_y P(y|x)\, ln \frac{P(y|x)}{q_x(y)} \quad (1)$$

Cette quantité est toujours supérieure ou égale à zéro, et est égale à zéro si et seulement si q_x est égale à$P(.|x)$. La minimisation de la divergence de Kullback-Leibler est donc un critère raisonnable pour obtenir une fonction q_x qui approxime P(·|x). Vu que le but est d'obtenir une bonne approximation pour toutes les valeurs de x qui pourraient être générées par P, il est logique de considérer la minimisation du critère

$$C(q) = E_X[D_{KL}(P(.|x)||q_x)]$$

$$= \int_x \sum_y P(x)P(y|x) \ln \frac{P(y|x)}{q_x(y)} dx \quad (2)$$

$C(q)$ Est ici ce que l'on appelle l'erreur de généralisation, c'est-à-dire. L'erreur moyenne sur des exemples tirés de P. Puisque P est inconnue, on minimise en pratique un critère $\hat{C}$ défini

de la même manière en remplaçant P par $\hat{P}$. C'est le principe de minimisation du risque empirique [8], et $\hat{C}$ s'écrit ici [26] :

$$\hat{C}(q) = \sum_{i=1}^{n} \frac{1}{n} \ln \frac{1}{q_{x_i}(y_i)} = -\frac{1}{n} \sum_{i=1}^{n} \frac{1}{n} \ln q_{x_i}(y_i) \quad (3)$$

Qui est appelée la *log-vraisemblance négative* (en anglais NLL, pour « Negative Log-Likelihood »). Ce critère est minimisé dès que $q_{x_i}(y_i) = 1$, pour tous les exemples d'entraînement $(x_i, y_i) \in$ D *et ce quelle que soit la valeur de* $q_x(y)$ *pour des valeurs de x non observées* dans D [26].

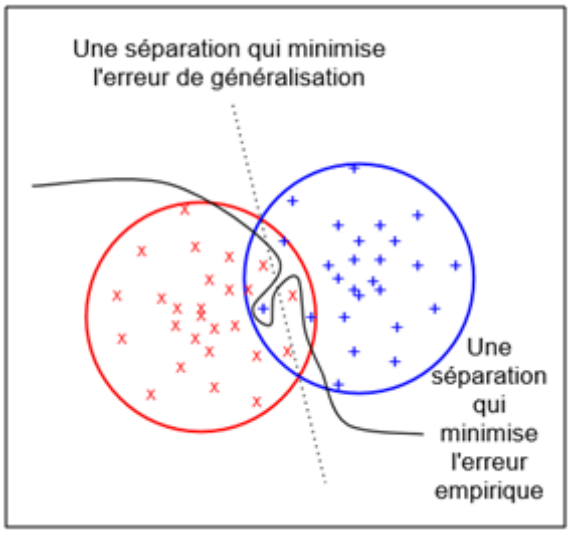

Figure 4.4. Situation de sur-apprentissage. : classification binaire (les classes sont les cercles rouge et bleu, avec respectivement les × et les + comme exemples d'entraînement). Une séparation idéale en termes d'erreur de généralisation serait la ligne en pointillés, mais un algorithme dont le but est uniquement de minimiser l'erreur empirique pourrait par exemple séparer les exemples selon la ligne pleine : la classification des exemples d'entraînement serait parfaite, mais l'erreur de généralisation serait plus élevée que celle de la ligne en pointillés [26].

Une fonction q peut donc minimiser $\hat{C}(q)$ sans nécessairement minimiser $C(q)$ Si c'est le cas, on est en situation de sur-apprentissage, illustrée en figure 1.4. Pour une distribution fixe des données, deux facteurs principaux augmentent le risque de sur-apprentissage [26] :

– Le manque d'exemples d'entraînement : moins il y a d'exemples, plus il existe de fonctions minimisant le critère $\hat{C}(q)$(éq. 3), parmi lesquelles seulement un petit nombre seront vraiment proches de la "vraie" solution au problème.

– Pas assez de contraintes dans la forme de la fonction q : moins la classe de fonctions à laquelle q appartient est restreinte, plus l'algorithme d'apprentissage risque de tirer

parti de la fléxibilité de q pour apprendre des "détails" des exemples d'entraînement, qui ne se généralisent pas à la vraie distribution *P*. C'est le cas par exemple dans la figure 1.4 où la séparation tarabiscotée des exemples par la ligne pleine permet de minimiser l'erreur empirique, mais va mener à une plus grande erreur de généralisation qu'une simple ligne droite [26].

3.6.2 Régularisation

Un moyen de lutter contre le sur-apprentissage est d'utiliser une technique dite de régularisation. Il existe plusieurs méthodes de régularisation, mais elles partagent le même principe : rajouter au processus d'apprentissage des contraintes qui, si elles sont appropriées, vont améliorer les capacités de généralisation de la solution obtenue.

Reprenons par exemple le cas de la classification, où l'on cherche à minimiser le critère $C(q)$(éq. 2), que l'on approxime par $\hat{C}(q)$(éq. 3). Comme nous venons de le voir, ce problème est mal défini car il existe une infinité de fonctions qui minimisent C, sans donner aucune garantie sur la valeur de C. Une première façon de régulariser le problème est donc de restreindre la forme de q : par exemple $x \in \mathbb{R}^d$et $y \in \{0,1\}$ on peut se limiter aux fonctions de la forme [26].

$$q_x(1) = \frac{1}{1+e^{-w^T x}} \tag{4}$$

Où $w \in \mathbb{R}^d$ est le vecteur de paramètres du modèle. Notons que si les x_i de l'ensemble d'entraînement sont linéairement indépendants, alors cette contrainte sur la forme de q n'est pas suffisante, puisqu'il est toujours possible que $\hat{C}(q)$soit arbitrairement proche de zéro sans pour autant avoir de garantie sur la valeur de $C(q)$. Une technique classique de régularisation consiste alors à rajouter au critère $\widehat{C}$ une mesure qui pénalise la *complexité* de la solution, suivant le principe du rasoir d'Occam qui dit que les hypothèses les plus simples sont les plus vraisemblables voir [3]. Une possibilité est de minimiser [26].

$$\tilde{C}(q) = \hat{C}(q) + \lambda||w||^2 \tag{5}$$

Au lieu de $\hat{C}$, pour q défini comme dans (l'éq. 4), afin d'empêcher le vecteur w de contenir des valeurs arbitrairement grandes (en valeur absolue). Le paramètre λ contrôle la force de cette contrainte (lorsque $\lambda \to +\infty$ la seule solution possible est la fonction constante $q_x(1) = q_x(0) = 0{,}5$, qui est la plus simple qu'on puisse imaginer). Le critère empirique $\hat{C}(q^*)$pour la fonction q^* qui minimise le critère régularisé $\tilde{C}$ pourrait ne pas être proche de

zéro, mais on peut souvent ainsi – pour certaines valeurs de λ – obtenir des valeurs plus basses du critère de généralisation C (celui qui nous intéresse vraiment). C'est le principe de la minimisation du risque structurel [8] [26].

Dans cet exemple, nous avons utilisé $||w||^2$ pour mesurer la complexité de la fonction *q*définie à partir de *w* par (l'éq. 4). En général, il n'existe pas une seule mesure de complexité universelle qui soit appropriée pour tous les algorithmes d'apprentissage, et le choix de la mesure de complexité à pénaliser joue un rôle très important. La complexité de Kolmogorov [7][4], est une mesure de complexité très générique qui est intéressante en théorie, même si en pratique elle est souvent impossible à utiliser directement. Elle consiste à dire que la complexité d'une fonction est la taille du plus petit programme qui l'implémente. Un premier obstacle à l'utilisation de cette complexité est le fait qu'il faille choisir un langage de programmation approprié : par exemple si le langage choisi contient une fonction primitive qui calcule le produit scalaire, alors, dans notre exemple ci-dessus la plupart des fonctions q définies par (l'éq. 4) ont la même complexité de Kolmogorov. Par contre, si le produit scalaire n'est pas une primitive du langage (et qu'il n'y a pas d'instruction de boucle), alors il faut l'écrire comme une somme de produits et q est d'autant plus complexe que w a d'éléments non nuls. Une autre difficulté est qu'il n'est en général pas possible d'optimiser la complexité de Kolmogorov de manière efficace, ce qui rend vaine son utilisation directe dans un processus d'optimisation. Elle a malgré tout de nombreuses applications, comme décrit dans le livre de [5] [26].

3.6.3 Malédiction de la dimensionnalité

On peut observer empiriquement – et dans certains cas justifier mathématiquement –que plus la dimension d de l'entrée x est élevée, plus les tâches d'apprentissage machine ont tendance à être difficiles à résoudre. C'est ce qu'on appelle *la malédiction de la dimensionalité* [1]. Il existe plusieurs manifestations de cette malédiction. La plus importante dans le contexte de cette thèse est le fait que le nombre de combinaisons possibles des entrées augmente exponentiellement avec la dimensionnalité d : en notant x_{ij} la valeur associée à la j-ème dimension de l'entrée x_i, si l'on suppose que ces entrées ne peuvent prendre qu'un

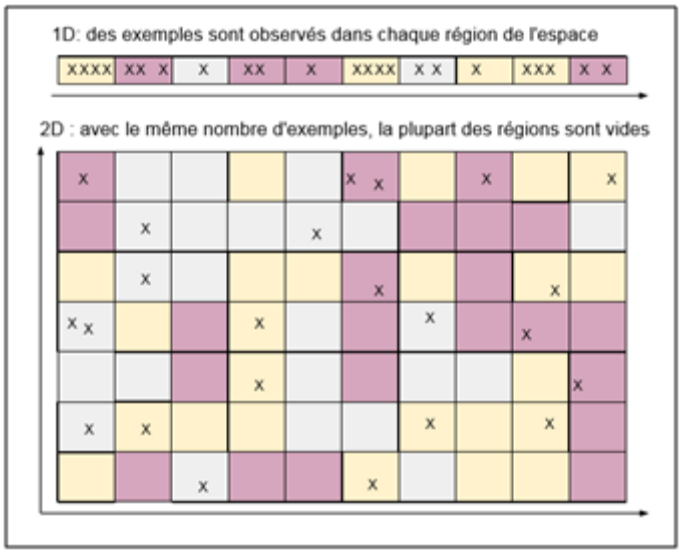

Figure 4.5. Malédiction de la dimensionalité : si l'algorithme partitionne l'espace en régions indépendantes, le nombre d'exemples nécessaires pour remplir ces régions augmente de manière exponentielle avec la dimension. Ici, la couleur d'une région représente la classe majoritaire dans cette région, et un tel algorithme pourrait bien généraliser à partir de 23 exemples d'entraînement pour le problème du haut (1D), mais pas pour celui du bas (2D) [26].

nombre fini*k* de valeurs, alors le nombre de combinaisons possibles est égal à k^d. Un algorithme qui apprend "bêtement" à associer une valeur à chaque combinaison sans partager d'information entre les différentes combinaisons n'a aucune chance de fonctionner en haute dimension, car il ne pourra pas généraliser aux multiples combinaisons qui n'ont pas été vues dans l'ensemble d'entraînement. Dans le cas où x_{ij} n'est pas contraint dans un ensemble fini de valeurs, l'intuition reste la même pour certains algorithmes qui consistent à "partitionner" $\mathbb{R}^d$ en régions indépendantes (possiblement de manière implicite) : si le nombre de ces régions augmente exponentiellement avec *d*, alors un tel algorithme aura de la difficulté à généraliser pour de grandes valeurs de *d*. La figure 2.5 illustre ce phénomène en une et deux dimensions, et il faut garder à l'esprit que la situation peut s'avérer encore bien pire lorsque l'on manipule des entrées à plusieurs centaines de dimensions [26].

3.7 **Différents types de modèles**

3.7.1 Modèles paramétriques

En apprentissage machine, un modèle paramétrique est défini par un ensemble Θ de paramètres de dimension fini, et l'algorithme d'apprentissage associé consiste à trouver la meilleure valeur possible de Θ. La figure 1.6 montre un exemple de régression linéaire en une dimension, sans régularisation.

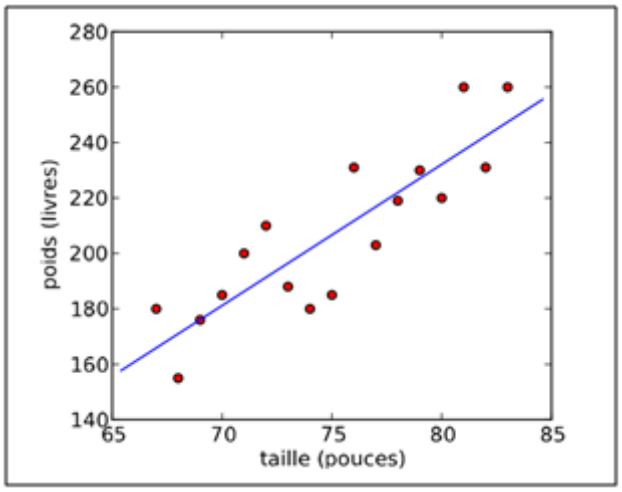

Figure 4.6. Modèle paramétrique : la régression linéaire. Les données (points rouges) sont un sous-ensemble des mêmes données que dans la figure 1.2, et la tâche est ici de prédire le poids d'un joueur de baseball en fonction de sa taille [26].

Les modèles paramétriques peuvent également être statistiquement inefficaces. S'il y a moins d'exemples d'apprentissage, alors le problème est sur-paramétré et on risque le sur-apprentissage : le modèle pourrait apprendre des paramètres taillés "sur mesure" pour les données d'entraînement, mais qui mèneront à une mauvaise généralisation. La conséquence de cette observation est qu'en général, un modèle avec un grand nombre de paramètres est statistiquement inefficace [26].

3.7.2 Modelés non paramétriques

Un modèle non paramétrique n'a au contraire pas d'ensemble exacte de paramètres : le nombre de variables utilisées par le modèle augmente généralement avec le nombre d'exemples dans l'ensemble d'entrainement. Un exemple de modèle non paramétrique pour résoudre le même problème de régression que celui décrit en 4.1 est l'algorithme des fenêtres de Parzen, aussi appelé régression de Nadaraya-Watson [6][9]. La figure 1.7 montre un exemple de régression par fenêtres de Parzen en une dimension [26].

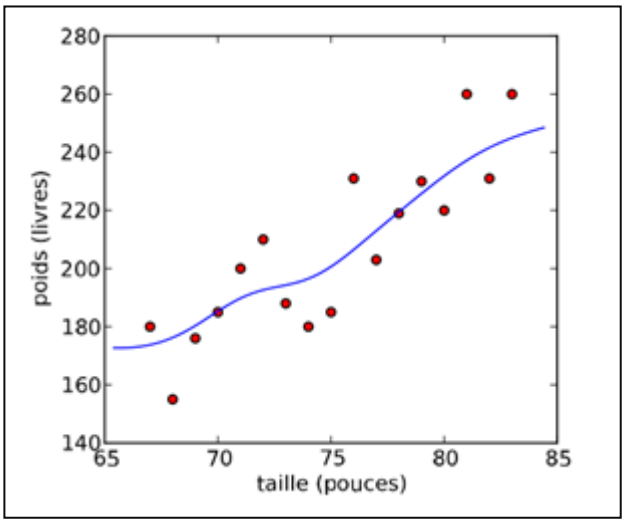

Figure 4.7. Modèle non paramétrique : régression par fenêtres de Parzen. Les données sont les mêmes que dans l'exemple de la régression linéaire (La figure 1.6) [26].

3.8 Algorithmes d'apprentissage

3.8.1 Réseaux de neurones

Comme leur nom l'indique, les réseaux de neurones sont inspirés de l'architecture du cerveau, étant organisés en couches de neurones connectées entre elles. La première couche, appelée la couche d'entrée, est de la même dimension que les entrées *x* et l'activation de son j-ème neurone (c'est-à-dire la valeur qu'il calcule) est égale à x_{ij} lorsque l'on calcule la prédiction du réseau sur l'exemple x_i. La dernière couche, appelée la couche de sortie, est de la même dimension que l'étiquette dans le cas d'une tâche supervisée. Par exemple, pour la classification, le j-ème neurone de la couche de sortie va calculer $P(Y = j|x_i)$ lorsque x_i est dans la couche d'entrée. Les couches intermédiaires sont appelées les couches cachées. Il existe de nombreuses variations dans les architectures de réseaux de neurones. L'architecture la plus connue est celle où il existe une unique couche intermédiaire h qui calcule une transformation non linéaire des entrées, de la forme [26] :

$$H(x) = \sigma(W^T x + b) \quad (7)$$

Où W est une matrice de poids, b est un vecteur de biais (de la même taille que le nombre de neurones dans la couche cachée), et σ est une transformation non linéaire élément par élément, dont l'opération sur chaque élément est typiquement la sigmoïde ou la tangente hyperbolique [26]

$$\text{Sigmoid(u)} = \frac{1}{1+e^{-u}} \quad (8)$$

$$\tanh(u) = \frac{e^{u}-e^{-u}}{e^{u}+e^{-u}} \quad (9)$$

La fonction donnant la sortie $\hat{y}$ en fonction de h dépend de la tâche ; en classification, on écrit souvent le $j^{éme}$ neurone de $\hat{y}$, qui estime $P(Y = j|x)$,sous la forme :

$$\frac{e^{V_j^T h+c_j}}{\sum_k e^{V_k^T h+c_k}} \quad (10)$$

Avec V_j la j-ème rangée d'une matrice de poids V, et c le biais du j-ème neurone de sortie. La figure 1.8 montre une telle architecture à une couche cachée.

L'apprentissage dans un réseau de neurones consiste à optimiser les poids et biais de façon à minimiser un coût approprié à la tâche. Par exemple, pour la classification, on minimisera la log-vraisemblance négative empirique (éq. 3) qui, en notant Θ les paramètres à optimiser, et $f_j(x_i; \Theta)$ la valeur du j-éme neurone de sortie lorsque x_i est en entrée du réseau, se réécrit [26] :

$$\hat{C}(\Theta) = -\frac{1}{n}\sum_{i=1}^{n} \ln f_{y_i}(x_i; \Theta) \quad (11)$$

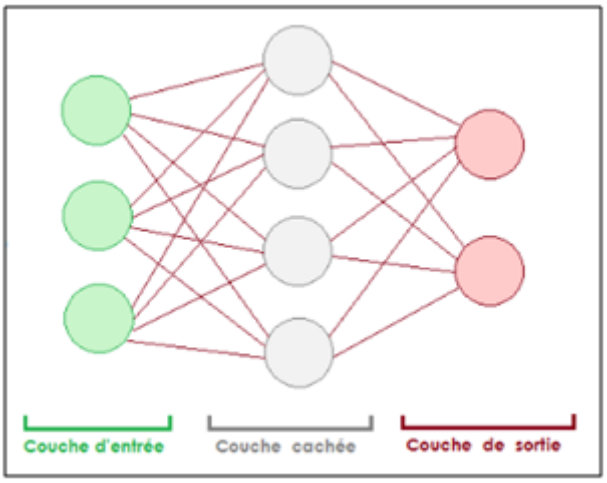

Figure 4.8. Réseau de neurones à une couche cachée. Une flèche du neurone i vers le neurone j indique que l'activation de j dépend directement de celle de i [26].

Un autre exemple, en apprentissage non supervisé, est celui des réseaux de neurones auto-associateurs dont le but est d'extraire dans la couche cachée une représentation qui

permet de reconstruire l'entrée x_i (donc l'étiquette y_i est en fait égale à x_i). Dans ce cas, le coût le plus fréquemment utilisé est l'erreur de reconstruction quadratique moyenne [26]

$$\hat{C}(\Theta) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}(f_i(x_i;\Theta) - x_{ij})^2 \quad (12)$$

Mais lorsque les entrées $x_{ij} \in \{0,1\}$, on peut également minimiser l'entropie croisée

$$\hat{C}(\Theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}((x_{ij} ln f_i(x_i;\Theta) + (1 - x_{ij})\ln(1 - f_i(x_i;\Theta) \quad (13)$$

Où le terme dans la somme peut s'interpréter comme la log-vraisemblance des données d'entraînement selon la distribution $P_\Theta(X_{ij} = 1|x_i) = f_i(x_{ij};\Theta)$.

L'apprentissage dans un réseau de neurones consiste à optimiser les poids et biais de façon à minimiser un coût approprié à la tâche. Les algorithmes d'optimisation les plus souvent utilisés pour minimiser $\hat{C}(\Theta)$sont des algorithmes d'optimisation locale basés sur l'idée de *la descente de gradient* : le gradient du coût $\hat{C}(\Theta)$par rapport aux paramètres Θ, noté $\frac{\partial \hat{C}(\Theta)}{\partial \Theta}$ indique la direction dans laquelle le coût augmente le plus lorsque Θ varie. Descendre le gradient signifie déplacer les paramètres Θ dans la direction opposée, de manière à diminuer le coût. Le gradient par rapport à tous les paramètres du réseau peut se calculer de manière efficace grâce à l'algorithme de rétro-propagation du gradient [21] [26].

En apprentissage non supervisé, on parle des réseaux de neurones auto-associateurs dont le but est d'extraire dans la couche cachée une représentation qui permet de reconstruire l'entrée x_i (donc l'étiquette $y_i$est en fait égale à x_i). Dans ce cas, le coût le plus fréquemment utilisé est l'erreur de reconstruction quadratique moyenne. On trouve donc que les réseaux de neurones représentent l'algorithme le plus approprié pour notre projet et pour l'implémentation [26].

3.8.2 Machines de Boltzmann restreintes

Dans sa version la plus simple, une machine de Boltzmann restreinte (RBM)est un algorithme non supervisé qui modélise la distribution $P(X)$ où $x \in \{0,1\}^d$ comme la marginale d'une distribution jointe [26]

$$P(x,h) = \frac{e^{-\varepsilon(x,h)}}{\mathrm{Z}} \quad (14)$$

Où Z est *la fonction de partition* assurant que cette probabilité est bien normalisée :

$$Z = \sum_{x,h} e^{-\varepsilon(x,h)} \quad (15)$$

Le vecteur $h \in \{0,1\}^k$ représente la couche cachée, tandis que le vecteur x est appelée la couche visible. Les éléments d'une couche sont généralement appelés unités plutôt que neurones. La quantité $\varepsilon(x,h)$ est *l'énergie* de la paire (x,h), et l'équation 13 montre que plus l'énergie est faible, plus cette paire est probable. Une forme typique pour ε est [26]:

$$\varepsilon(x,h) = h^T W x - x^T b - h^T c \qquad (16)$$

Où la matrice $W \in \mathbb{R}^{k\times d}$ et les vecteurs b $\in\mathbb{R}^d$ et c $\in\mathbb{R}^k$ sont les paramètres du modèle. Bien que ce modèle ait originellement été introduit sous un autre nom [44], il s'agit bien d'une forme particulière de machine de Boltzmann [16][15]. Comparée à une machine de Boltzmann générique, l'énergie d'une RBM à la propriété que les probabilités conditionnelles $P(x|h)$ et $P(h|x)$ se factorisent, ce qui rend l'entraînement (un peu) plus aisé. Cette propriété se visualise lorsque l'on représente une RBM sous la forme d'un modèle graphique non dirigé [25], comme dans la figure 1.9 : il y a des connexions entre les unités visibles et les unités cachées, mais aucune connexion de visible à visible, ni de cachée à cacher [26].

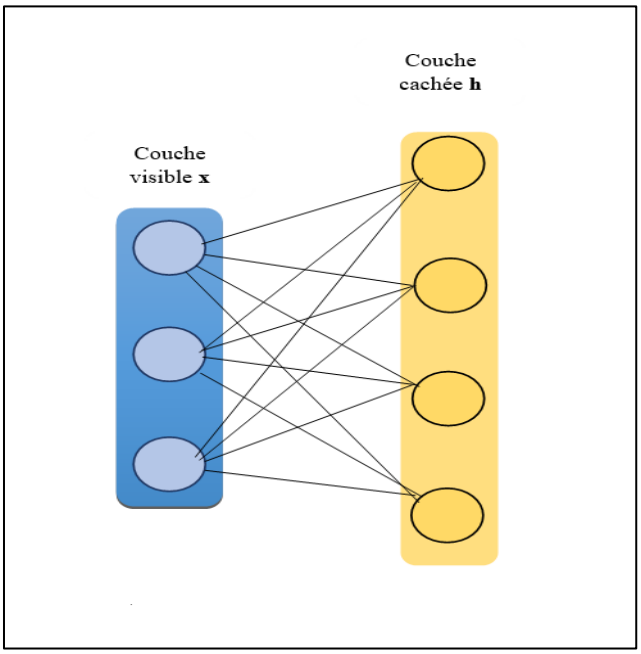

Figure 4.9. Machine de Boltzmann restreinte [26].

3.8.3 Architectures profondes (Deep Learning)

Le terme d'architecture profonde désigne toute une famille de modèles inspirés des réseaux de neurones, dont le point commun est la composition de transformations successives, permettant de calculer une fonction complexe de l'entrée. Par exemple, un réseau de neurones avec une seule couche intermédiaire peut être considéré comme une architecture profonde si l'on rajoute des couches cachées : chaque couche additionnelle augmente la profondeur du

réseau, et déterminer la profondeur idéale en fonction des données fait partie de la problématique des algorithmes d'apprentissage pour architectures profondes. Les réseaux de ce type ont longtemps été ignorés, d'une part parce qu'ils se sont avérés beaucoup plus difficiles à optimiser que les réseaux à une seule couche cachée, d'autre part parce qu'il a été démontré que les réseaux à une seule couche sont des approximateurs universels [17] [26]. L'intérêt pour les réseaux profonds est récemment réapparu après avoir découvert qu'une initialisation non supervisée des poids du réseau peut mener à de bien meilleures performances que l'initialisation aléatoire utilisée jusqu'à présent.

3.8.4 K-plus proches voisins

L'algorithme des k plus proches voisins est un algorithme non paramétrique utilisé pour la régression et la classification. Etant donnée une mesure de distance dans l'espace d'entrée $\mathbb{R}^d$ (souvent prise comme la distance Euclidienne$||x_i - x_j||$) la prédiction du modèle sur un exemple de test $x \in T$ dépend uniquement des k plus proches voisins de x dans l'ensemble d'entraînement D. En notant $i_1(x), \ldots, i_k(x)$ les indices des k exemples de D les plus proches de x selon la distance choisie (ses "voisins"), la prédiction du modèle en régression est la moyenne des étiquettes observées chez ces k voisins [26] :

$$f(x) = \frac{1}{k}\sum_{j=1}^{k} y_{i_j(x)} \quad (17)$$

Et en classification il s'agit d'un vote parmi les k voisins :

$$f(x) = argmax_y \sum_{j=1}^{k} 1_{y=y_{i_j(x)}} (18)$$

Où en cas d'égalité parmi les votes le modèle choisit aléatoirement l'une des classes majoritaires. La classification par les k plus proches voisins est illustrée en figure 1.11.

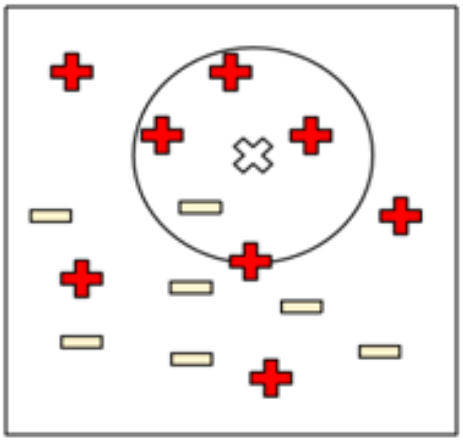

Figure 4.10. K-plus proches voisins. (k = 5, tâche de classification) [26].

3.8.5 Fenêtres de Parzen

L'algorithme des fenêtres de Parzen a déjà été présenté dans le contexte de la régression non paramétrique, où on l'appelle parfois la régression à noyau ou la régression de Nadaraya et Watson [6][9]. On peut également utiliser une approche similaire en apprentissage non supervisé pour l'estimation de densité [20] [18], en estimant la densité de probabilité au point *x* par [26] :

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} K(x, x_i) \tag{19}$$

Ce qui correspond à placer une masse de probabilité "autour" de chaque exemple d'apprentissage x_i, dans un volume défini par le noyau K. Ici, K doit respecter les contraintes [26]

$$K(x, x_i) \geq 0$$

$$\int_x K(x, x_i)dx = 1 \tag{20}$$

De manière à ce que *f* soit une densité de probabilité valide. Le choix le plus répandu pour K est le noyau Gaussien, mais avec la normalisation appropriée [26]:

$$K(x, x_i) = \mathcal{N}\left(x_i; x_j, \sigma^2\mathbf{I}\right) = \frac{1}{(2\pi)^{d/2}\sigma d} e^{-\frac{||x_i - x_j||^2}{2\sigma^2}} \tag{21}$$

où l'on note $\mathcal{N}(\cdot; \mu, \Sigma)$ la densité de probabilité d'une Gaussienne de moyenne μ et covariance Σ. Un exemple d'estimation de densité par fenêtres de Parzen avec un noyau Gaussien est montré en figure 1.12 [26].

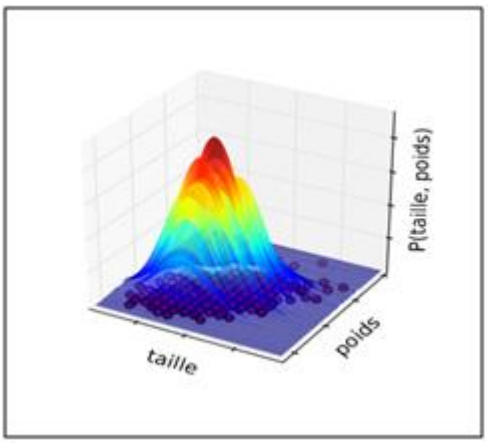

Figure 4.11. Fenêtres de Parzen pour l'estimation de densité [26].

3.8.6 Mélanges de Gaussiennes

Les mélanges de Gaussiennes généralisent les fenêtres de Parzen (avec noyau Gaussien) pour l'estimation de densité, en écrivant la densité comme une somme pondérée de Gaussiennes [52] :

$$f(x) = \sum_{j=1}^{N} \alpha_j \mathcal{N}(x; \mu_j, \Sigma_j) \quad (22)$$

Où N est le nombre de composantes du mélange, et α_j le poids de la j^{ème} composante (les poids sont tels que $\alpha_j \geq 0$ et$\Sigma_j \alpha_j = 1$). L'interprétation dite "générative'' de cette équation est que le modèle suppose que chaque exemple observé a été généré de la façon suivante [26] :

1.Une composante j est choisie aléatoirement, avec probabilité α_j.

2.Un exemple est généré par une distribution Gaussienne centrée en μ_j avec covariance Σj.

Si N = n, $\alpha_j = n^{-1}$, $\mu_j = x_i$ et $\Sigma_j = \sigma^2\mathbf{I}$ on retrouve les fenêtres de Parzen non paramétriques vues précédemment. Mais on peut également apprendre un modèle paramétrique de mélange en fixant N et en optimisant les poids α_j, les centres μ_j et les covariances Σ_j de chaque Gaussienne. L'algorithme le plus populaire pour l'apprentissage d'un mélange de Gaussiennes est l'algorithme *Espérance-Maximisation* (EM) [14].

3.8.7 Méthodes à noyau

Plusieurs algorithmes mentionnés précédemment utilisent une fonction noyau $K(x_i, x_j)$ pour mesurer la similarité entredeux entrées x_i et x_j. Les méthodes dites "à noyau" incluent en particulier une catégorie d'algorithmes basés sur ce que l'on appelle "l'astuce du noyau", qui s'applique à tout algorithme qui peut s'exprimer uniquement à partir de produits scalaires de la forme $x_i^T x_j$. Cette astuce consiste à remplacer $x_i^T x_j$ dans l'algorithme d'origine par une fonction noyau $K(x_i, x_j)$, où K doit satisfaire certaines propriétés mathématiques (on dit que le noyau est défini positif – c'est le cas par exemple du noyau Gaussien que nous avons déjà utilisé). Cela revient à appliquer l'algorithme sur les données transformées implicitement et de manière non linéaire par une fonction ϕ telle que $\varphi(x_i)^T \varphi(x_j) = K(x_i, x_j)$ (une telle fonction ϕ existe automatiquement si K est défini positif, et en pratique on n'a pas besoin de la calculer explicitement). L'intérêt du noyau est principalement d'effectuer une extraction de caractéristiques non linéaire à partir des entrées, ce qui peut améliorer les performances de l'algorithme [26].

La plus célèbre méthode à noyau exploitant cette astuce est l'algorithme de la machine à vecteurs de support (SVM, pour "Support Vector Machine" en anglais). Il s'agit d'un algorithme de classification basé sur l'idée de marge : pour obtenir une meilleure généralisation, il est préférable de laisser une marge entre les exemples et la surface de décision [11][13][24]. Ce concept de marge est illustré par la figure 1.17 : dans le cas linéaire (c'est à dire. Sans utiliser l'astuce du noyau) la marge du classifeur est la distance entre l'hyper-plan séparant les deux classes et les exemples les plus proches. Dans le cas non linéaire, le même principe s'applique dans l'espace des exemples projetés implicitement par ϕ, ce qui se traduit dans l'espace des entrées par une séparation non linéaire des exemples de chaque classe. Depuis les SVMs, l'astuce du noyau et l'idée de marge ont inspiré un grand nombre de nouveaux algorithmes, ainsi que la "noyautisation" d'algorithmes existants [22] [26].

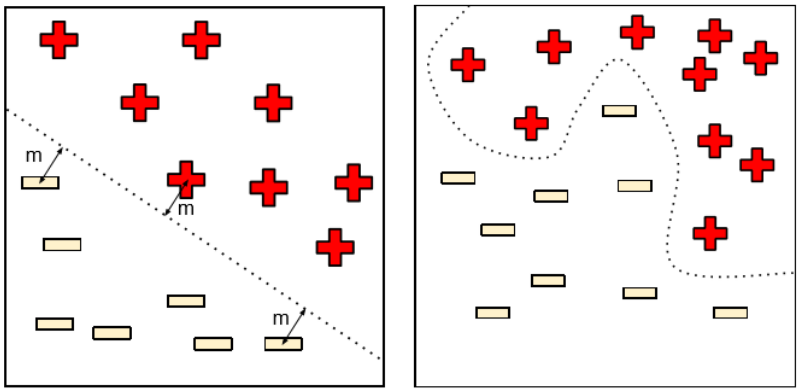

Figure2.13.. Machine à vecteurs de support, dans le cas linéaire (à gauche) et non linéaire (à droite) [26].

3.8.8 Arbres de décision

Les arbres de décision forment une famille d'algorithmes d'apprentissage utilisés pour la classification et la régression [12]. Un arbre de décision est un arbre où chaque nœud interne représente un test sur l'entrée x_i. L'exemple typique d'un arbre de décision est un arbre binaire où le test effectué à chaque nœud k est de la forme $x_{ij}<\theta_k$, c'est-à-dire. Qu'on compare la $j^{éme}$ coordonnée de x_ià un seuil θ_k : si elle est plus petite, on continue de parcourir l'arbre en suivant la première branche du nœud, sinon on suit la seconde branche. Lorsqu'on atteint finalement une feuille de l'arbre (un nœud sans enfants), on dit que l'exemple x_i appartient à cette feuille. La figure 1.18 montre un tel arbre de décision [26].

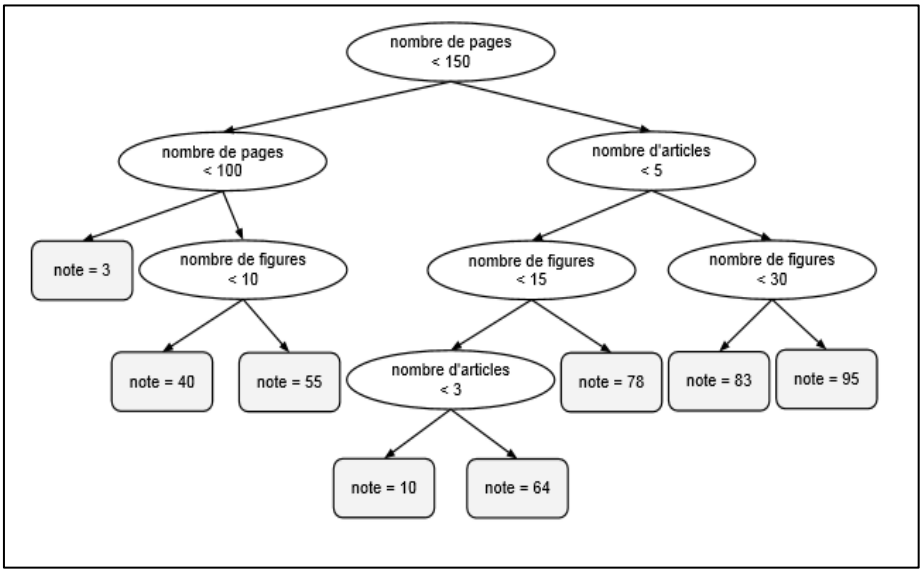

Figure 2.18. Arbre de décision typique [26]

L'apprentissage de ce type d'arbre de décision consiste à choisir les variables testées à chaque nœud, les seuils de comparaison, la profondeur de l'arbre, ainsi que la fonction de décision associée à chaque feuille. Il existe plusieurs algorithmes pour cela, mais leur principe de base est de faire en sorte que le résultat du test effectué à chaque nœud donne de l'information supplémentaire sur $P(Y|x)$.Idéalement, la distribution $P(Y|x)$pour un nouvel exemple *x* doit être bien approximée par la distribution empirique des étiquettes des exemples d'entraînement appartenant à la même feuille que *x* [26].

3.8.9 Méthodes Bayésiennes

Les méthodes Bayésiennes tirent leur nom de la célèbre règle de Bayes [26] :

$$P(\Theta|\mathcal{D}) = \frac{P(\Theta|\mathcal{D})P(\Theta)}{P(\mathcal{D})} \tag{28}$$

Qui est ici appliquée avec d'une part les paramètres Θ d'un modèle, et d'autre part les données d'entraînement D observées. La quantité $P(\Theta|D)$ est appelée la vraisemblance : c'est la probabilité d'observer les données D en supposant qu'elles ont été générées par notre modèle dont les paramètres sont Θ. P(Θ) est appelée la distribution a priori : c'est une distribution sur l'espace des paramètres qui reflète notre croyance sur les valeurs possibles des paramètres avant l'observation des données. P(Θ|D) est alors calculée comme étant proportionnelle au produit de la vraisemblance par la distribution a priori : elle est appelée la distribution a posteriori, c'est-à-dire. Qu'elle indique la probabilité des paramètres après avoir observé les données. Le terme P(D) est un terme de normalisation qui peut se calculer, si

nécessaire, par $P(D) = \int_{\Theta} P(D|\Theta)P(\Theta)d\Theta$. Il suffira de garder à l'esprit qu'il s'agit de méthodes probabilistes, qui ont l'avantage en particulier de prendre en compte l'incertitude de manière naturelle : par exemple, la prédiction d'un modèle Bayésien supervisé sur une nouvelle entrée *x* peut s'écrire comme [26]

$$P(y|x, D) = \int_{\Theta} P(\Theta)P(\Theta|D)d\Theta \quad (29)$$

Et la variance de cette distribution peut être interprétée comme l'incertitude sur la prédiction de l'étiquette *y*. Notons qu'il n'est en général pas trivial de calculer une telle intégrale, et que plusieurs techniques ont été développées spécifiquement dans ce but [10] [26].

3.9 Conclusion

Dans ce chapitre nous avons défini la notion de l'apprentissage artificiel et ses caractéristiques et nous avons introduit ses composants. Nous avons appris ses avantages. Ainsi, nous avons présenté quelques travaux antérieurs et récents ayant un lien avec notre travail de recherche.

Dans le prochain chapitre, Nous expliquant la conception générale de notre système de détection de texte, ainsi les approches utilisées et on montre les résultats obtenus.

Chapitre 03 : Conception et implémentation.

3.3 Introduction

Ce dernier chapitre est consacré à la conception et la mise en place de notre projet qui permettra de détecter le texte dans les images de scènes naturelles en utilisant une méthode appartient à la famille de techniques de l'apprentissage profond basé sur l'alogothme de YOLO. Dans ce chapitre nous allons décrire également les différentes parties de notre système, les détails relatifs à chaque phase ainsi que l'explication et les détails de chaque étape de ces phases.

3.4 Conception générale de notre système

Généralement, le processus de détection de texte dans les images comporte ces étapes : Tout d'abord, un cadre d'ancrage pivoté avec des informations d'angle est utilisé comme cadre de délimitation du texte sur diverses orientations. Deuxièmement, des caractéristiques de différentes échelles sont extraites de l'image d'entrée pour déterminer la probabilité, la confiance et les cadres de délimitation inclinés du texte. Enfin, l'intersection de distance de rotation sur la suppression non maximale de l'union est utilisée pour éliminer la redondance et acquérir les résultats de détection avec la plus grande précision.

3.5 Conception détaillé

Dans cette partie, Notre modèle qui basé sur l'architecture de l'algorithme YOLOv3 est décrite en détail.

YOLO est un algorithme de détection basé sur un apprentissage profond, et l'un des algorithmes de détection d'objets les plus rapides. Bien qu'il ne s'agisse plus de l'algorithme de détection d'objets le plus précis, c'est un très bon choix lorsque vous avez besoin d'une détection en temps réel, sans perte de précision trop importante. Il détermine les cadres de délimitation inclinés du texte dans une image de scène naturelle et les classe dans un seul cadre unifié. Plus précisément, nous avons proposée une série d'améliorations basées sur YOLOv3, qui amélioré l'algorithme de régression de la boîte englobante et repensé la fonction de perte du cadre, afin qu'il puisse réaliser la détection de texte de manière flexible dans les scènes naturelles. Le flux de traitement des données est présenté dans la figure 3.1.

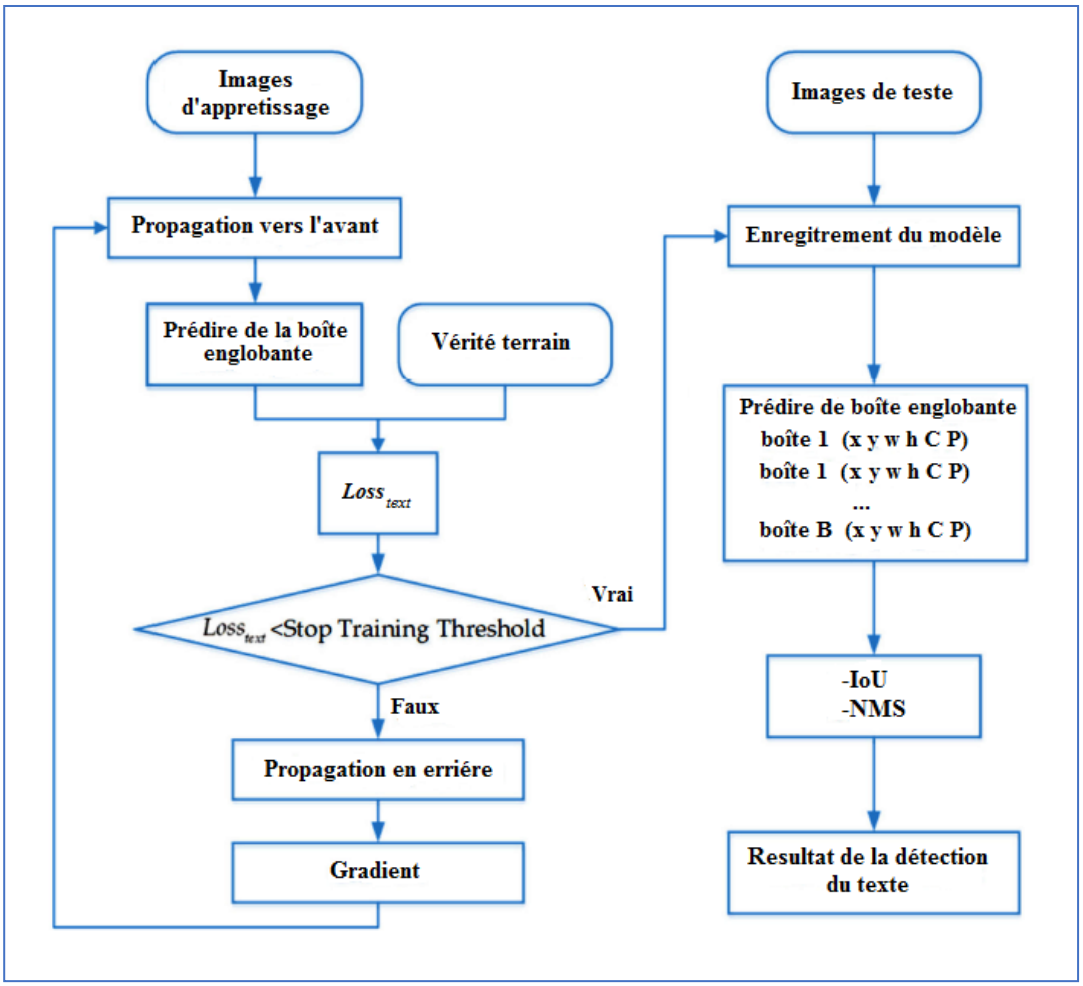

Figure 3.1. Notre système de détection de texte dans une image.

3.3.1 Architecture de YOLOv3

Inspiré des architectures ResNet et FPN (Feature-Pyramid Network), l'extracteur de fonctionnalités YOLOv3, appelé Darknet-53 (il a 52 convolutions) contient des connexions de saut (comme ResNet) et 3 têtes de prédiction (comme FPN), chacune traitant l'image à une compression spatiale différente. Comme son prédécesseur, le YOLOv3 offre de bonnes performances sur une large gamme de résolutions d'entrée. Dans le zoo modèle de GluonCV, vous pouvez trouver plusieurs points de contrôle: chacun pour une résolution d'entrée différente, mais en fait les paramètres réseau stockés dans ces points de contrôle sont identiques. Testé avec une résolution d'entrée de 608x608 sur l'ensemble de validation COCO-2017, le YOLOv3 a obtenu un score de 37 mAP. Ce score est identique à la version d'apprentissage de GluonCV Faster-RCNN- ResNet50 , (une architecture plus rapide que RCNN qui utilise ResNet50 comme son épine dorsale), mais 17 fois plus rapide. Dans ce modèle de zoo, les seuls détecteurs assez rapides pour rivaliser avec le YOLOv3 (architectures Mobilenet-SSD) ont obtenu un mAP de 30 et moins.

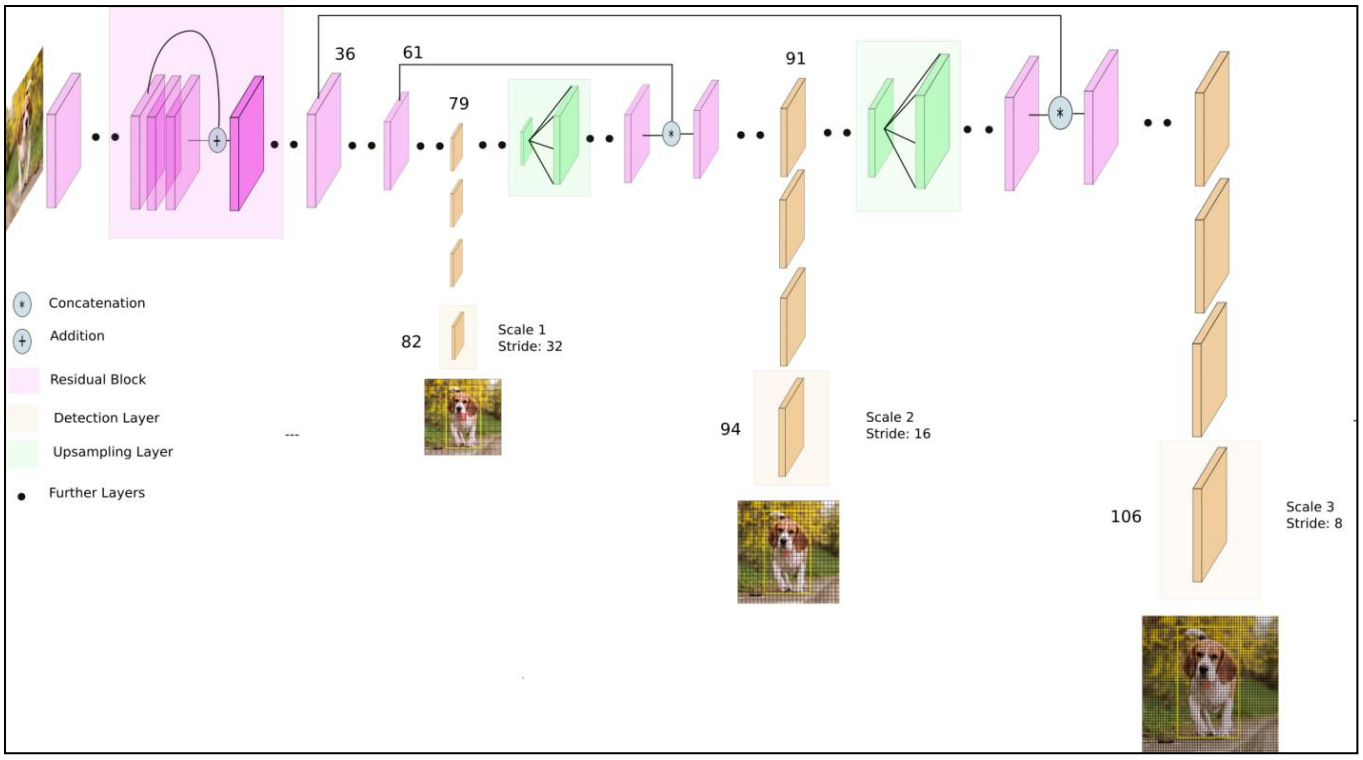

Figure 5.2. Architecture de YOLOv3, Pour obtenir une détection finé, trois branches sont utilisées pour la détection d'objets, l'échelle de la carte des caractéristiques de chaque branche étant différente. Chaque branche prédit la confiance, la probabilité de classe et les cadres de délimitation inclinés du texte.

3.3.2 Darknet-53

Tout d'abord, YOLO v3 utilise une variante de Darknet, qui dispose à l'origine d'un réseau de 53 couches formé sur Imagenet. Pour la tâche de détection, 53 couches supplémentaires sont empilées dessus, ce qui nous donne une architecture sous-jacente entièrement convolutive de 106 couches pour YOLO v3. C'est la raison de la lenteur de YOLO v3 par rapport à YOLO v2. Voici à quoi ressemble maintenant l'architecture de YOLO.

3.3.3 Détection à trois échelles

La nouvelle architecture se vante de connexions de saut résiduelles et de sur échantillonnage. La caractéristique la plus marquante de la v3 est qu'elle effectue des détections à trois échelles différentes. YOLO est un réseau entièrement convolutif et sa sortie éventuelle est générée en appliquant un noyau 1 x 1 sur une carte de caractéristiques. Dans YOLO v3, la détection se fait en appliquant des noyaux de détection 1 x 1 sur des cartes de caractéristiques de trois tailles différentes à trois endroits différents du réseau.

La forme du noyau de détection est 1 x 1 x (B x (5 + C)). Ici, B est le nombre de boîtes englobantes qu'une cellule sur la carte d'entités peut prédire, «5» est pour les 4 attributs de boîte englobante et une confiance d'objet, et c'est le nombre de classes. Dans YOLO v3 entraîné sur COCO, B = 3 et C = 80, donc la taille du noyau est 1 x 1 x 255. La carte de caractéristiques produite par ce noyau a la même hauteur et la même largeur que la carte de caractéristiques précédente, et a des attributs de détection le longdu profondeur comme décrit ci-dessus.

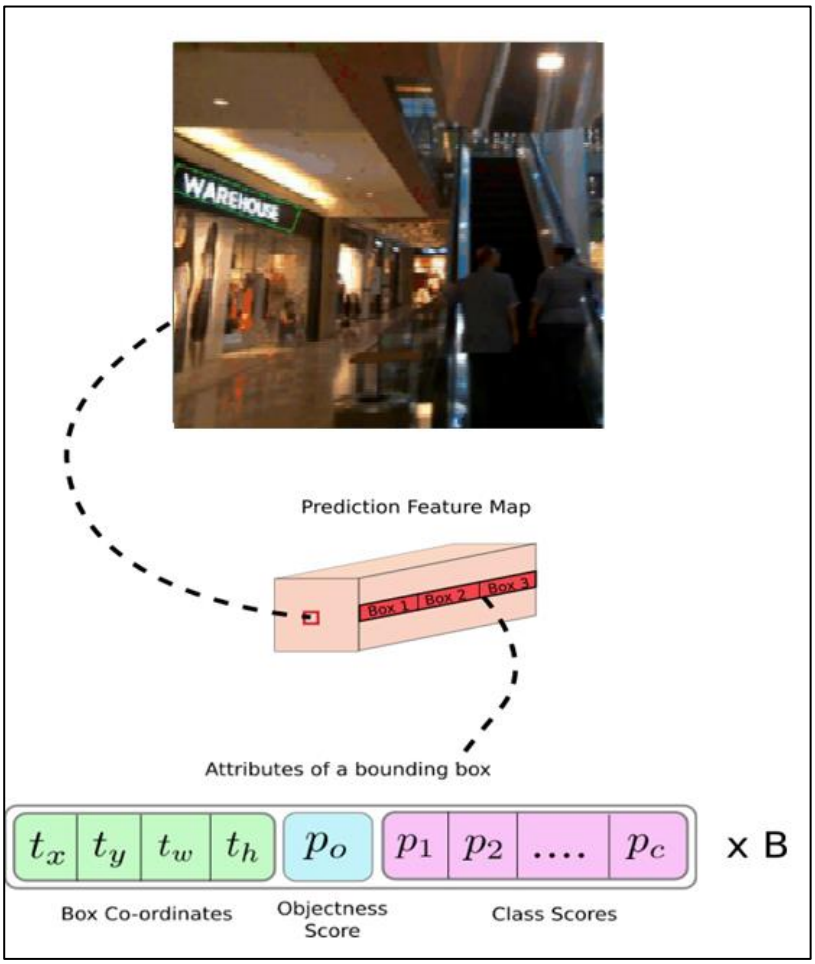

Figure 3.3. Grille d'images.

3.3.4 Meilleure détection d'objets plus petits

Les détections à différentes couches permettent de résoudre le problème de la détection de petits objets, une plainte fréquente avec YOLO v2. Les couches sur échantillonnées concaténées avec les couches précédentes aident à préserver les caractéristiques à grain fin qui aident à détecter les petits objets.

La couche 13 x 13 est responsable de la détection des grands objets, tandis que la couche 52 x 52 détecte les objets plus petits, la couche 26 x 26 détectant les objets moyens.

Voici une analyse comparative de différents objets sélectionnés dans le même objet par différentes couches.

3.3.5 Représentation de la boîte englobante inclinée

Dans la phase d'apprentissage, la vérité terrain d'un ensemble de régions de texte est représentée par (x,y,w,h,θ), où les coordonnées (x,y) sont exprimées comme les coordonnées du point central de vérité terrain dans l'image système de coordonnées, comme illustré à la figure 3.. La stratégie pour déterminer w, h et est la suivante : le bord où l'axe des x tourne dans le sens inverse des aiguilles d'une montre et d'abord parallèle à l'axe des x est défini comme w. L'angle entre la baguette et l'axe des x est θ et la plage de θ est comprise entre $(-90^\circ, 0^\circ)$. Le côté perpendiculaire à l'axe est noté cendre, donc la valeur de n'est pas nécessairement plus petite que h. Il y a trois avantages à utiliser la stratégie de représentation ci-dessus. En raison de la périodicité de l'angle est éliminé. Deuxièmement, il est pratique d'effectuer l'opération de régression de la boîte englobante inclinée. Troisièmement, par rapport à la représentation traditionnelle à 8 points d'une boîte englobante inclinée (x1, y1, x2, y2, x3, y3, x4, y4), cette représentation peut calculer facilement la nouvelle valeur de vérité terrain après la rotation des images d'entraînement.

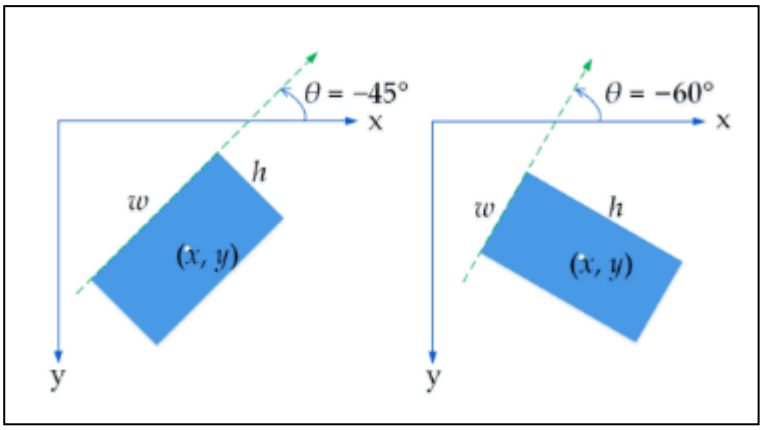

Figure 3.4. Représentation de la boîte englobante inclinée.

3.3.6 Boîte d'ancrage de rotation

Comme la boîte de vérité du texte est étiquetée à l'aide d'une boîte rectangulaire avec un angle de rotation, la boîte horizontale d'ancrage traditionnelle, représentée uniquement par les paramètres d'échelle et de rapport hauteur/largeur, ne convient pas à la détection de texte dans les scènes naturelles. Par conséquent, nous concevons les ancres de rotation (ancres R) en ajustant plusieurs paramètres. Premièrement, les échelles des boîtes d'ancrage sont conçues pour être de 8, 16 et 32 pixels. Deuxièmement, comme les zones de texte ont généralement

des échelles différentes, nous définissons trois proportions de 2:1, 5:1 et 8:1 pour couvrir les lignes de texte avec plusieurs proportions. De plus, un paramètre d'orientation est ajouté pour contrôler la direction de l'ancrage. Quatre orientations différentes, à savoir 0◦, −30◦, −60◦ et −90◦ sont utilisées pour garantir que l'angle a la valeur initiale optimale pendant le processus d'apprentissage. La stratégie d'ancrage est résumée à la figure 3.4. Après les étapes de représentation des données ci-dessus, une ancre R est générée avec cinq paramètres (x, y, w, h, θ).

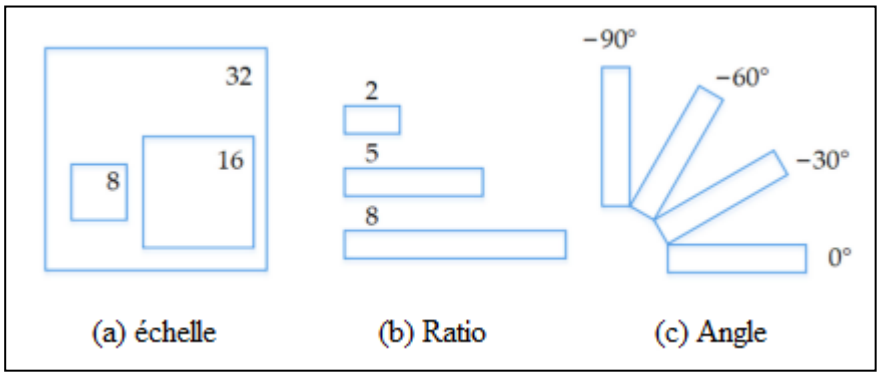

Figure 3.5. R-ancre défini dans notre cadre.

3.3.7 Plus de cadres de délimitation par image

Pour une image d'entrée de même taille, YOLO v3 prédit plus de cadres de délimitation que YOLO v2. Par exemple, avec sa résolution native de 416 x 416, YOLO v2 prédit 13 x 13 x 5 = 845 boîtes. À chaque cellule de la grille, 5 boîtes ont été détectées à l'aide de 5 ancres.

D'autre part, YOLO v3 prédit des boîtes à 3 échelles différentes. Pour la même image de 416 x 416, le nombre de boîtes prédites est de 10 647. Cela signifie que YOLO v3 prédit 10 fois le nombre de boîtes prédit par YOLO v2. Vous pouvez facilement imaginer pourquoi il est plus lent que YOLO v2. À chaque échelle, chaque grille peut prédire 3 cases à l'aide de 3 ancres. Puisqu'il y a trois échelles, le nombre de boîtes d'ancrage utilisées au total est de 9, 3 pour chaque échelle.

3.3.8 Apprentissage de la détection de texte

Dans YOLOv3, la fonction de perte est définie comme la somme de perte de classification des objets, de perte de confiance et de perte de régression de la boîte englobante.

$$L_{text} = L_{box} + L_{confidense} + L_{class} \quad (1)$$

La perte de confiance et de classification sont définies comme :

$$L_{confidense} = -\sum_{i=0}^{S\times S}\sum_{j=0}^{B} I_{ij}^{obj}[\hat{C}_i \log(C_i) + (1-\hat{C}_i)\log(1-\hat{C}_i)] - \lambda_{noobj}\sum_{i=0}^{S\times S}\sum_{j=0}^{B} I_{ij}^{noobj}[\hat{C}_i \log(C_i) + (1-\hat{C}_i)\log(1-\hat{C}_i)] \quad (2)$$

$$L_{class} = -\sum_{i=0}^{S\times S} I_{ij}^{obj} \sum_{c\epsilon classes} [\hat{P}_i(c)\, log(P_i(c)) + (1-\hat{P}_i(c))\, log(1-P_i(c))] \quad (3)$$

Dans l'équation (2), S×S représente le nombre de cellules dans la carte des caractéristiques. B est le nombre de prédicteurs dans chaque grille. I_{ij}^{obj} représente s'il y a une cible qui tombe dans la j-ème boîte englobante de la i ème cellule de grille. λ_{noobj} fait référence aux paramètres d'équilibrage qui contrôlent le trade-off entre ces termes. $\hat{C}_i$ et C_i désignent la confiance vraie et prédite, respectivement. Dans l'équation (3). I_{ij}^{obj} indique si la cible apparaît dans la cellule i. $\hat{P}_i(c)$ se réfère à la vraie probabilité de la cible, tandis que $P_i(c)$ fait référence à la valeur prédite. Dans le code officiel de YOLOv3, deux types de perte de régression de la boîte englobante sont implémentés : La perte MSE et la perte CIoU, nous implémentons une régression de boîte aux limites inclinée basée sur la perte MSE. Étant donné le paramètre d'angle de la boîte englobante inclinée, la complexité de calcul de l'angle gradient augmente au cours de la propagation en arriére de la fonction de perte CIoU.

La perte MSE est définie comme :

$$L_{box} = \lambda_{coord}\sum_{i=0}^{S\times S}\sum_{j=0}^{B} I_{ij}^{obj}\,(2-\hat{w}_i\times\hat{h}_i)[(x_i-\hat{x}_i)^2+(y_i-\hat{x}_i)^2] + \lambda_{coord}\sum_{i=0}^{S\times S}\sum_{j=0}^{B} I_{ij}^{obj}\,(2-\hat{w}_i\times\hat{h}_i)\left[(w_i-\hat{w}_i)^2+\left(h_i-\hat{h}_i\right)^2\right] \quad (4)$$

Ici, λ_{coord} est un paramètre d'équilibrage avec la valeur fixée à 1, I_{ij}^{obj} indique si non l'objet cible tombe dans le j-ème cadre de délimitation de la i-ème cellule de grille. (x_i, y_i, w_i, h_i) et $(\hat{x}_i, \hat{y}_i, \hat{w}_i, \hat{h}_i)$ se représente la coordonnée du centre, hauteur et largeur de la boîte englobante prédite et du la vérité terrain, respectivement.

La perte CioU est définie comme :

$$L_{box} = 1 - IoU + \frac{P^2(b,b^{gt})}{c^2} + \alpha v \tag{5}$$

$$\alpha = \frac{\vartheta}{(1-\mathrm{IoU})+\vartheta} \tag{6}$$

$$v = \frac{4}{\pi^2} * \left(arctan\frac{\mathrm{w}^{\mathrm{gt}}}{\mathrm{h}^{\mathrm{gt}}} - \arctan\frac{\mathrm{w}}{\mathrm{h}}\right)^2 \tag{7}$$

Ici, IoU est l'intersection sur l'union entre la boîte prédite et la Vérité terrain. $P^2(b, b^{gt})$ est la distance euclidienne. c est la diagonale du plus petit boîte enveloppante couvrant les boîtes englobantes. (w, h) et $(\mathrm{w}^{\mathrm{gt}}, \mathrm{h}^{\mathrm{gt}})$ représente la hauteur et largeur de la boîte prédite et la vérité terrain, respectivement.

Nous ajoutons une branche de perte angulaire basée sur la fonction de perte MSE, concevoir la perte fonction de la régression de la boîte englobante inclinée. Étant donné une boîte d'ancrage de rotation $A = (a_x, a_y, a_w, a_h, a_\theta)$ et sa boîte de vérité terrain correspondante $G = (g_x, g_y, g_w, g_h, g_\theta)$ notre objectif est d'apprendre un mappage f telle que $f(A) = P$

Où $P = (p_x, p_y, p_w, p_h, p_\theta)$ est la boîte englobante prédite et $P \approx G$ La définition de la relation de mappage entre A et P est exprimée comme suit :

$$\begin{cases} p_x = a_w d_x(A) + a_x \\ p_y = a_h d_y(A) + a_y \\ p_w = a_w exp\big(d_w(A)\big) \\ p_h = a_h exp\big(d_h(A)\big) \\ p_\theta = d_\theta(A) + a_\theta \end{cases} \tag{8}$$

Ici $d_x(A)$ et $d_y(A)$ désignent la transformation invariante d'échelle des deux centres entre A et P. $d_\theta(A)$ représente la transformation invariante de l'angle. Et $d_w(A)$, $d_h(A)$ caractérisent la transformation d'échelle exponentielle de la largeur et de la hauteur respectivement.
Comme illustré à la figure 3.5.

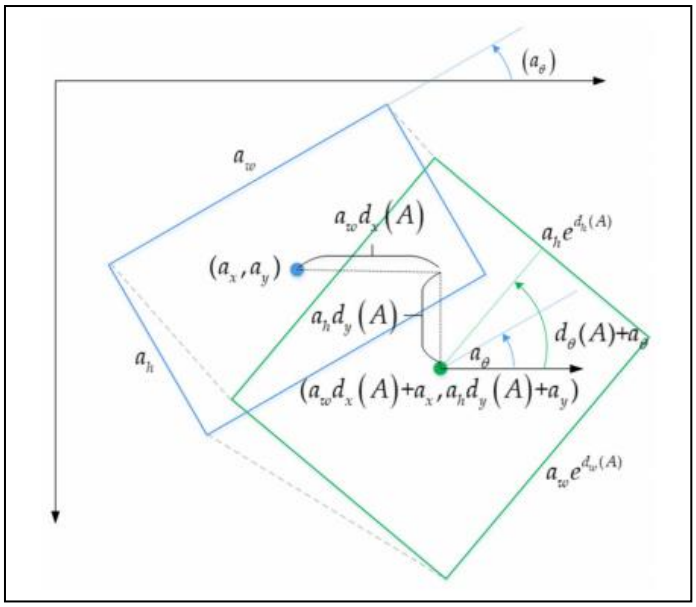

Figure 5.1. Illustration de la transformation entre la boîte d'ancrage de rotation et la boîte englobante prédite. Le rectangle bleu représente la boîte d'ancrage de rotation et le rectangle vert représente le cadre de délimitation.

Le but de la régression de la boîte englobante inclinée est d'entraîner un ensemble de paramètres W pour faire $Y = WX$. Pendant le processus d'apprentissage, l'entrée X est la carte des caractéristiques de chaque boîte d'ancrage à la place de $(a_x, a_y, a_w, a_h, a_\theta)$, la carte des caractéristiques est représentée par $\varphi(A)$, et Y est calculé par la vérité terrain G et la région d'ancrage de rotation A pour obtenir la translation et le zoom, exprimé par t_* où $*$ est l'un de (x, y, w, h, θ) comme :

$$\begin{cases} t_x = (g_x - a_x)/a_w \\ t_y = (g_y - a_y)/a_h \\ t_w = \log(g_w/a_w) \\ t_h = \log(g_h/a_h) \\ t_x = g_\theta - a_\theta \end{cases} \quad (9)$$

Grâce à un entraînement itératif, W fait le $W\varphi(A) \approx t$ avec notre fonction de perte acquise comme :

$$L_{box} = \lambda_{coord} \sum_{i=0}^{S\times S} \sum_{j=0}^{B} I_{ij}^{obj} (2 - h_i \times w_i) \left(t_*^i - \mathrm{w}_*^T \varphi(\mathrm{A})\right)^2 \quad (10)$$

Ici λ_{coord} fait référence aux paramètres d'équilibrage avec la valeur fixée à 1. w_i et h_i dans $(2 - h_i \times w_i)$ sont la largeur et la hauteur de la vérité terrain, respectivement. Le rôle de $(2 - h_i \times w_i)$ est responsable de l'équilibrage de la valeur de perte générée lors de la détection de grandes et petits objets.

3.4 La base de données

Nous avons sélectionné une base de données contenant du texte directionnel : ICDAR2015 [58] pour des expériences visant à évaluer les performances sur divers textes directionnels. Pour démontrer d'avantage la polyvalence de notre modèle. Une brève description de la base de données pertinent est donnée ci-dessous.

- **ICDAR2015 :** La base de données de texte de scène ICDAR2015 provient du défi 4 du concours de lecture robuste ICDAR2015. La base de données comprend 1000 images d'entraînement et 500 images de test, qui ont été capturées à l'aide de lunettes Google avec des résolutions relativement faibles. Les annotations d'instance de texte ont quatre sommets, qui forment une boîte englobante quadrilatérale irrégulière avec des informations d'orientation. [58]

3.5 Langage, Logiciels et librairies utilisés dans l'implémentation

Pour développer mon application, j'ai utilisé un environnement de programmation, et des outils et des librairies différentes pour la programmation en arrière-plan (back-end), exécutable sur plusieurs IDE (environment).

3.5.1 Python

Figure 3.6 Python logo.

Python est un langage de programmation de haut niveau interprété (il n'y a pas d'étape de compilation) et orienté objet avec une sémantique dynamique. Il est très sollicité par une large communauté de développeurs et de programmeurs. Python est un langage simple, facile à apprendre et permet une bonne réduction du cout de la maintenance des codes. Les bibliothèques (packages) python encouragent la modularité et la réutilisabilité des codes. Python et ses bibliothèques sont disponibles (en source ou en binaires) sans charges pour la majorité des plateformes et peuvent être redistribués gratuitement [57].

3.5.2 TensorFlow

Figure 5.7. TensorFlow logo.

TensorFlow est une bibliothèque de logiciels open source pour le calcul numérique haute performance. Son architecture flexible permet un déploiement facile du calcul sur une variété de plates-formes (CPU, GPU, TPU), et des ordinateurs de bureau aux clusters de serveurs en passant par les périphériques mobiles et périphériques.

Développé à l'origine par des chercheurs et des ingénieurs de l'équipe Google Brain au sein de l'organisation IA de Google, il est livré avec un support solide pour l'apprentissage automatique et l'apprentissage en profondeur et le noyau de calcul numérique flexible est utilisé dans de nombreux autres domaines scientifiques [52].

3.5.3 Flask

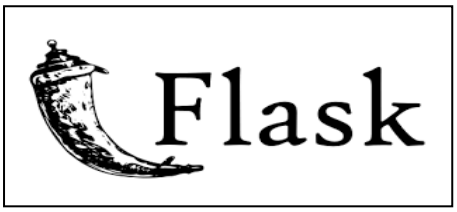

Figure 5.8. Flask logo.

Flask est un micro framework open-source de développement web en Python. Il est classé comme microframework car il est très léger. Flask a pour objectif de garder un noyau simple mais extensible. Il n'intègre pas de système d'authentification, pas de couche d'abstraction de

base de données, ni d'outil de validation de formulaires. Cependant, de nombreuses extensions permettent d'ajouter facilement des fonctionnalités.[4] Il est distribué sous licence BSD [62].

3.5.4 jQuery

Figure 5.9. jQuery logo.

est une bibliothèque JavaScript libre et multiplateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web [61].

3.5.5 Matplotlib

Figure 5.10. Matplotlib logo.

Matplotlib est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques. Elle peut être combinée avec les bibliothèques python de calcul scientifique NumPy et SciPy. Matplotlib est distribuée librement et gratuitement sous une licence de style BSD [60].

3.5.6 Keras

Figure 3.11 Keras logo

Keras est une API de réseaux de neurones de haut niveau, écrite en Python et capable de fonctionner sur TensorFlow. Il a été développé en mettant l'accent sur l'expérimentation rapide. Être capable d'aller de l'idée à un résultat avec le moins de délai possible est la clé pour faire de bonnes recherches. Il a été développé dans le cadre de l'effort de recherche du

projet ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), et son principal auteur et mainteneur est François Chollet, un ingénieur Google.

En 2017, l'équipe TensorFlow de Google a décidé de soutenir Keras dans la bibliothèque principale de TensorFlow. Chollet a expliqué que Keras a été conçue comme une interface plutôt que comme un cadre d'apprentissage end to end. Il présente un ensemble d'abstractions de niveau supérieur et plus intuitif qui facilitent la configuration des réseaux neuronaux indépendamment de la bibliothèque informatique de backend. Microsoft travaille également à ajouter un backend CNTK à Keras aussi [53].

3.5.7 NumPy

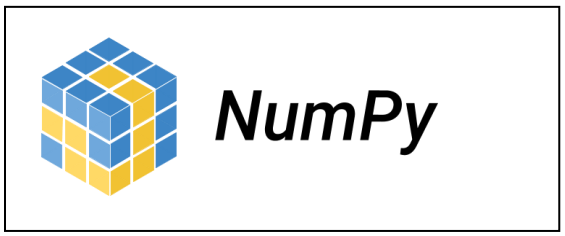

Figure 5.12. NumPy logo.

NumPy est une bibliothèque pour langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux [63].

3.5.8 PIL

Figure 5.13. Pillow logo.

Pillow est une bibliothèque de traitement d'image, qui est un fork et successeur du projet PIL (*Python Imaging Library*). Elle est conçue de manière à offrir un accès rapide aux données contenues dans une image, et offre un support pour différents formats de fichiers tels que PPM, PNG, JPEG, GIF, TIFF et BMP.

Pillow dispose de capacités de traitement d'images relativement puissantes, et a pour but d'offrir une solide base à toute application générale de traitement d'images [51].

3.5.9 OpenCV

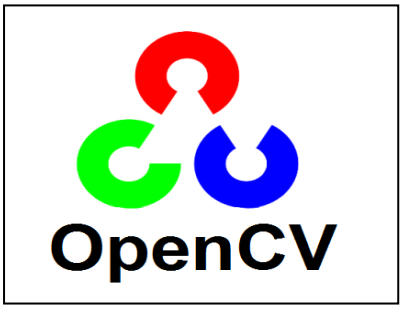

Figure 5.14. OpenCV logo.

OpenCV (pour Open Computer Vision) est une bibliothèque graphique libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel. La société de robotique Willow Garage et la société ItSeez se sont succédé au support de cette bibliothèque. Depuis 2016 et le rachat de ItSeez par Intel, le support est de nouveau assuré par Intel [56].Cette bibliothèque est distribuée sous licence BSD.

3.5.10 Tkinter

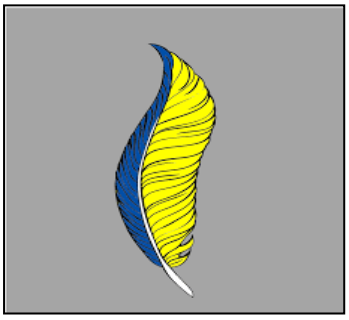

Figure 5.15. Tkinter logo.

Tkinter (de l'anglais Tool kit interface) est la bibliothèque graphique libre d'origine pour le langage Python, permettant la création d'interfaces graphiques. Elle vient d'une adaptation de la bibliothèque graphique Tk écrite pour Tcl [54].

3.5.11 PyCharm

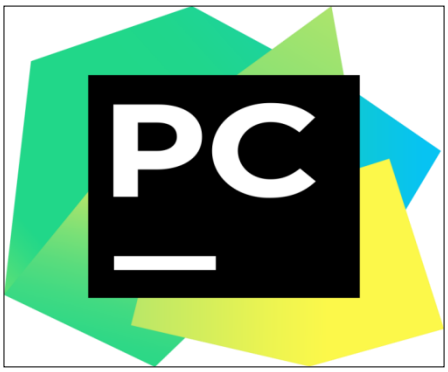

Figure 5.16. PyCharm logo.

Est un environnement de développement intégré utilisé pour programmer en Python.

Il permet l'analyse de code et contient un débogueur graphique. Il permet également la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement web avec Django.

Développé par l'entreprise tchèque JetBrains, c'est un logiciel multi-plateforme qui fonctionne sous Windows, Mac OS X et Linux [55].

3.5.12 Google colaboratory

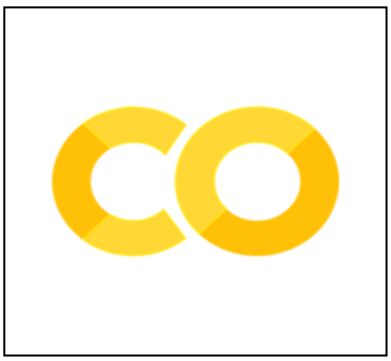

Figure 3.17. Google colaboratory logo.

Google colaboratory est une application web utilisée pour programmer dans plusieurs langages de programmation, dont Python, il m'a permis aussi d'utiliser les ressources disposées par GOOGLE dont NVIDIA Tesla K80 GPU pour la compilation et l'exécution du code en si peu de temps.

3.5.13 Configuration utilisée dans l'implémentation

Pour mettre en œuvre cette application, les matériaux ayant les caractéristiques suivantes ont été utilisés :

- Processeur : Intel(R) CoreTM i5-7200U CPU @2.50 GHZ
- RAM: 8.00 GB
- Système d'exploitation Windows 10.

3.6 Résultat

Dans ce que suit nous allons présenter les résultats obtenus pour la détection de texte dans les scènes naturelle en utilisant un système de détection à base de YOLOv3.

- Présentation graphique de mAP et l'erreur du modèle basé sur YOLOv3

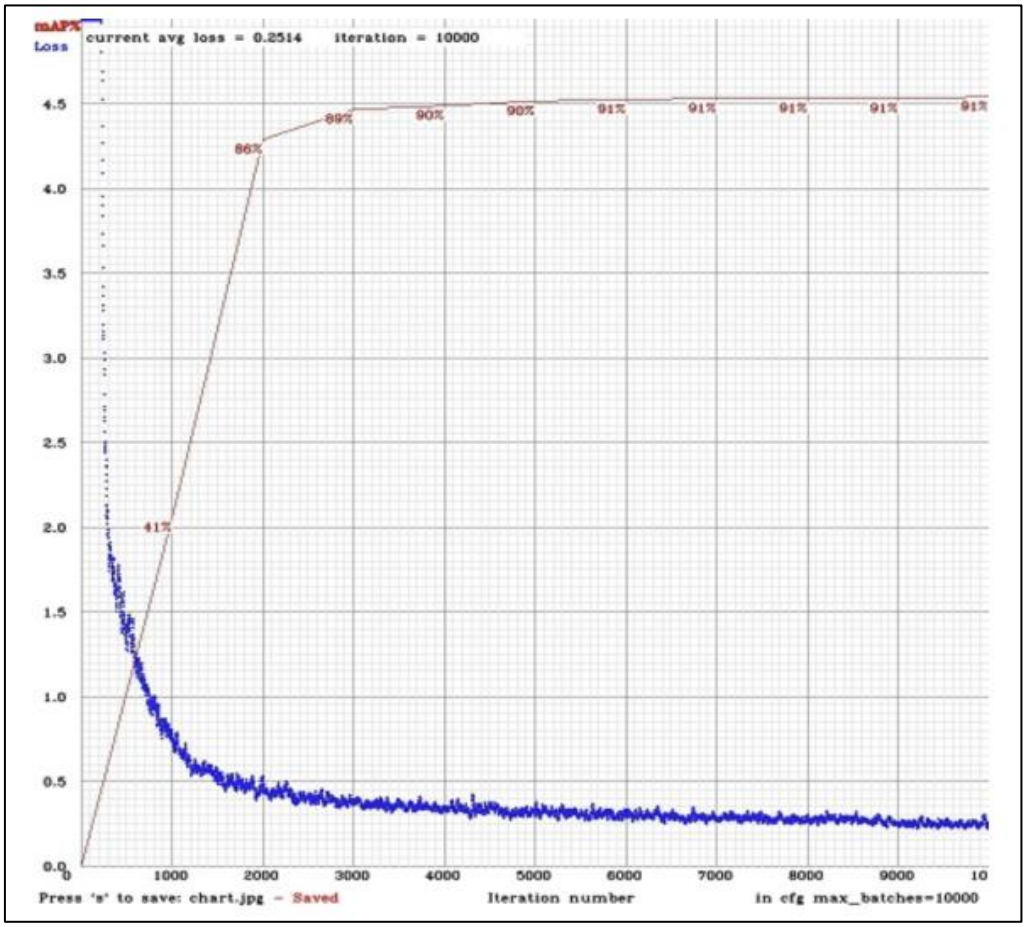

Figure 5.18. Présentation graphique de mAP et de l'erreur du modèle basé sur YOLOv3

3.6.1 Utilisation du modèle après la phase d'apprentissage

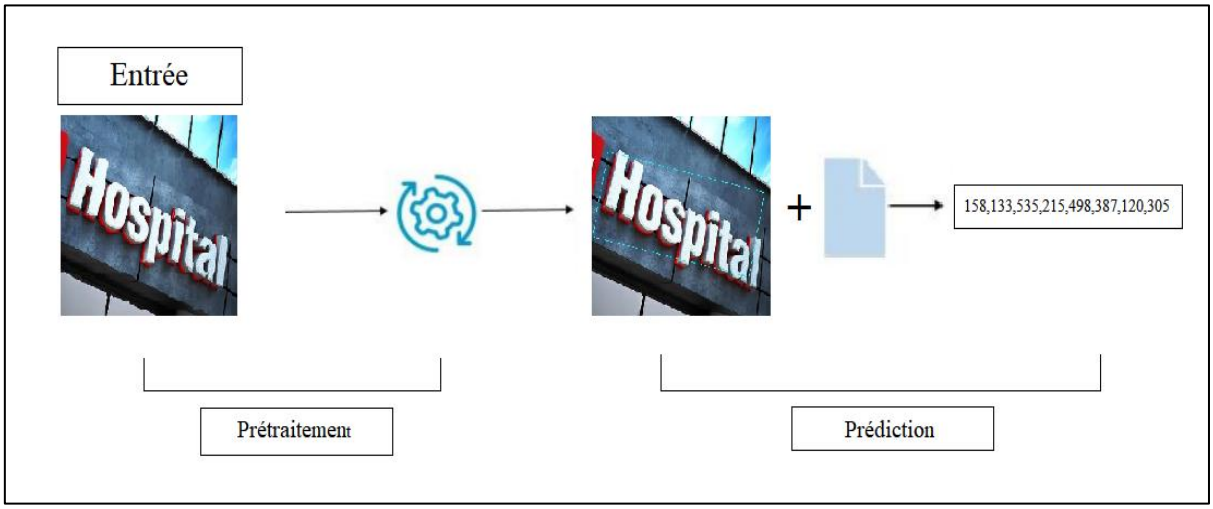

Figure 5.19. Exemple comment tester notre modèle.

Figure 5.20. Exemple sur une image.

3.6.2 Évaluation sur le référentiel de texte orienté

Nous évaluons notre modèle sur la base de données ICDAR2015. Le modèle est affiné pour 50 k itérations sur la base de données d'apprentissage de l'ICDAR2015. Pendant la phase de réglage, le taux d'apprentissage commence à partir de 1,0 × 10−3 et est multiplié par 1/10 après 4,0 × 104 et 4,5 × 104 itérations. Les résultats de détection de texte sont présentées à la figure 3.21.

Figure 5.21. Résultats de détection de texte sur certaines images de texte, Boîtes englobantes vertes : détections correctes.

3.6.3 Évaluation sur le repère de texte horizontal

Certains résultats de détection obtenus sont illustrés à la figure 3.22, qui montrent que notre méthode peut gérer convenablement la détection du texte horizontal dans des images naturelles.

Figure 5.21. Exemple des résultats du détection notre modèle, qui basé dur YOLOv3 sur des images contient texte horizontale, Boîtes englobantes vertes : détections correctes.

3.6.4 Évaluation sur le repère de textes multilingues

Certains résultats de détection obtenus sont illustrés à la figure 3.22, qui montrent que notre méthode peut gérer convenablement la détection de textes multilingues dans des images naturelles.

Figure 5.22. Exemple des résultats de détection notre modèle, qui basé dur YOLOv3 sur des images multilingues, Boîtes englobantes vertes : détections correctes.

3.6.5 Analyse et discussion

Notre modèle peut atteindre des vitesses plus élevées que les méthodes de pointe car notre réseau a deux avantages. Premièrement, la méthode proposée adopte Darknet53 comme épine dors a le réseau. Par rapport aux méthodes de détection existantes utilisant VGGNet ou ResNet comme backbone, un grand nombre de noyaux de convolution 1 × 1 sont exploités dans le réseau Darknet53 pour réduire les dimensions des cartes de caractéristiques, ce qui réduit le nombre de paramètres et la taille du modèle considérablement. Deuxièmement, par rapport à les méthodes basées sur la stratégie en deux étapes, notre modèle proposé basée sur la stratégie en une étape stratégie régresse la boîte englobante directement à partir des cartes de caractéristiques convolutives sans en s'appuyant sur la proposition de région, économisant ainsi le temps nécessaire au calcul de la proposition de région.

Deuxièmement, par rapport à l'algorithme NMS incliné traditionnel utilisé par R2CNN, une pénalité de distance, pour résoudre le problème de la fausse suppression causée par chevauchement des cadres de délimitation de différents textes. En particulier, dans la zone de texte dense, l'effet est plus satisfaisant. Troisièmement dans notre réseau, des astuces efficaces sont utilisées pour améliorer la capacité d'extraction de fonctionnalités réseau telles que SSP, PANet SAM, et leur efficacité a été vérifiée dans YOLOv3. L'effet combiné des raisons mentionnées ci-dessus rend notre modèle plus robuste que les méthodes concurrentes dans détection de texte orienté arbitrairement.

3.6.6 Limites de l'algorithme proposé

La méthode proposée surpasse les méthodes existantes de manière significative en termes d'efficacité de détection tout en maintenant une précision élevée, mais présente des limites dans la détection de texte de scène naturelle de petite taille. Il s'agit d'une limitation courante pour les objets détecteurs basés sur YOLOv3. Une autre limitation est que la méthode n'est pas bonne pour détecter le texte incurvé.

3.7 Conclusion

Nous avons présenté dans ce chapitre la conception et l’implémentation général et détailler de notre projet de fin d’études, qui consiste à la détection du texte dans des scènes naturelles où le texte peut être d'orientation arbitraire et d'échelles variées et qui basé sur YOLOv3.

Conclusion Générale

Les développements technologiques de ces vingt dernières années ont permis aux systèmes numériques d'envahir notre vie quotidienne. Il y a plus à démontrer que le numérique occupe aujourd'hui une place de choix dans le monde. Parmi les composantes majeures des systèmes numériques, une grande importance est accordée à l'image. La représentation et le traitement des images numériques font l'objet de recherches très actives à l'heure actuelle. Le traitement des images est un très vaste domaine qui a connu, et connaît encore, une évolution importante depuis quelques décennies.

Dans ce projet de master, nous avons concentré sur la détection de texte dans les images de scènes naturelles, en utilisant un apprentissage approfondi (deep learning) afin d'améliorer la précision. Pour cela nous avons proposé une série d'améliorations basées sur YOLOv3 pour permettre la détection du texte où le texte pourrait être orienté arbitrairement et à des échelles variées.

Ce travail de recherche avait pour but de concevoir et implémenter un modèle de détection du texte. Afin de sélectionner la boîte englobante inclinée optimale, nous avons proposé une methode, qui considère non seulement le facteur d'angle de la boîte englobante inclinée mais aussi la distance du point central entre les deux boîtes. De plus, les représentations de la boîte d'ancrage, de l'algorithme de régression de la boîte englobante et de la fonction de perte sont améliorées pour s'adapter à la détection du texte ayant subi une rotation arbitraire. Des comparaisons expérimentales et des analyses de modèles ont été menées sur la base de données ICDAR2015, sur cette base de données, notre modèle a atteint une F-mesure de 0.86. Les résultats montrent que le modèle proposé peut atteindre un niveau avancé de détection de texte avec une efficacité de calcul très élevée. Cependant, il peut encore être amélioré en termes de précision de détection. Premièrement, la dorsale du réseau peut être améliorée en utilisant un mécanisme d'attention avancé. Deuxièmement, la fonction de perte améliorée des boîtes englobantes inclinées basées sur la perte Pixels-IoU pourrait être envisagée.

Notre mémoire est organisé en trois chapitres avec une introduction générale et une conclusion générale, dans le premier chapitre on a parlé de la vision par ordinateur, dans le deuxième chapitre nous avons défini les notions de l’apprentissage automatique et ses caractéristiques, et le troisième la conception et la modélisation de notre système de détection.

En ce qui concerne nos perspectives pour les travaux futurs, notre idée et notre objectif seront d'améliorer l'ensemble du système de détection de texte avec quelques ajouts importants :

- Rendre notre système capable de pour détecter le texte incurvé.
- Ajouter un mode en ligne de détection de texte (détection en temps réel)
- Ajouter plus de fonctionnalités (version mobile, site web).
- Essayer de publier une publication scientifique concernant nos travaux, et pourquoi pas de rejoindre la liste de benchmarking.

Bibliographie

[1] K. E. F. D. a. A. T. Tike Judd, «“Learning to predict where humans look”,» Proceedings of IEEE 12th International Conference on Computer Vision, pp. 2106-2113, 2009.

[2] Bellman, R. 1961, Adaptive Control Processes: A Guided Tour, Princeton University Press, New Jersey.

[3] Blumer, A., A. Ehrenfeucht, D. Haussler et M. Warmuth. 1987, “Occam’srazor”, Inf. Proc. Let., vol. 24, p. 377–380.

[4] Kolmogorov, A. N. 1965, “Three approaches to the quantitative definition of information”, Problems of Information and Transmission, vol. 1, no 1, p. 1–7.

[5] Li, M. et P. Vitànyi. 2008, An Introduction to Kolmogorov Complexity and Its Applications, 3e ´ed., Springer, New York, NY.

[6] Nadaraya, E. A. 1964,“On estimating regression”, Theory of Probability and its Applications, vol. 9, p. 141–142.

[7] Solomonoff, R. J. 1964,“A formaltheoryof inductive inference”, Information and Control, vol. 7, p. 1–22, 224–254. Sutton, R. et A. Barto. 1998, Reinforcement Learning : An Introduction, MIT Press.

[8] Watson, G. S. 1964, “Smooth regression analysis”, Sankhya - The Indian Journal of Statistics, vol. 26, p. 359–372.

[9] Weizenbaum, J. 1966, “ELIZA-a computer program for the study of natural language communication between man and machine”, Commun. ACM, vol. 9, no 1, p. 36–45.

[10] Barber, D. 2011, Bayesian Reasoning and Machine Learning, Cambridge University Press.

[11] Boser, B. E., I. M. Guyon et V. N. Vapnik. 1992, “A training algorithm for optimal margin classifiers”, dans COLT ’92 : Proceedings of the fifth annual workshop on Computational learning theory,ACM,NewYork,NY, USA, p. 144–152.

[12] Breiman, L., J.H. Friedman, R.A. OlshenetC.J. Stone.1984, Classification and Regression Trees, Wadsworth International Group, Belmont, CA.

[13] Cortes, C. etV.Vapnik.1995, “Supportvectornetworks”, Machine Learning, vol. 20, p. 273–297.

[14] Dempster, A. P., N. M. Laird et D. B. Rubin. 1977, “Maximum-likelihood from incomplete data via the EM algorithm”, Journal of Royal Statistical Society B, vol. 39, p. 1–38.

[15] «Hinton, G. E. et T. J. Sejnowski. 1986, “Learning and relearning in Boltzmann machines”, dans Parallel Distributed Processing : Explorations in the Microstructure of Cognition,» . Volume 1 : Foundations, édité par D. E. Rumelhart et J. L. McClelland, MIT Press, Cambridge, MA, p. 282–317.

[16] Hinton, G. E., T. J. Sejnowski et D. H. Ackley. 1984,“Boltzmann machines : Constraint satisfaction networks thatlearn”, rapport technique TR-CMUCS-84-119, Carnegie-Mellon University, Dept. of Computer Science.

[17] Hornik, K., M. Stinchcombe et H. White. 1989, “Multilayer feedforward networks are universal approximators”, Neural Networks, vol. 2, p. 359– 366.

[18] Parzen, E. 1962, “On the estimation of a probability density function and mode”, Annals of Mathematical Statistics, vol. 33, p. 1064–1076.

[19] Rasmussen, C. E. et C. Williams. 2006, Gaussian Processes for Machine Learning, MIT Press.

[20] Rosenblatt, M. 1956,“Remarks on some nonparametric estimates of a density function”, The Annals of Mathematical Statistics, vol. 27, no 3, p. 832–837.

[21] Rumelhart, D. E., G. E. Hinton et R. J. Williams. 1986, “Learning representations by back-propagating errors”, Nature, vol. 323, p. 533–536.

[22] Scholkopf, B. et A. J. Smola. 2002, Learning with Kernels : Support Vector Machines, Regularization, Optimization and Beyond, MIT Press, Cambridge, MA.

[23] Smolensky, P. 1986, “Information processing in dynamical systems : Foundations of harmony theory”, dans Parallel Distributed Processing, vol. 1, édité par D. E. Rumelhart et J. L. McClelland, chap. 6, MIT Press, Cambridge, p. 194–281.

[24] Vapnik, V. 1998, Statistical Learning Theory, Wiley, Lecture Notes in Economics and Mathematical Systems, volume 454.

[25] Wainwright, M. J. et M. I. Jordan. 2008, “Graphical models, exponential families, and variational inference”, Foundations and Trends in Machine Learning, vol. 1, no 1-2, p. 1–305.

[26] ‘’Apprentissage machine efficace : théorie et pratique’’, par Olivier Delalleau , Département d’informatique et de recherche opérationnelle Faculté des arts et des sciences.

[27] Watanabe, S. Watanabe, “Pattern recognition: Human and Mechanical”, New York Wiley, 1985.

[28] Theodoridis et Koutroumbas, S. Theodoridis, K. Koutroumbas, “Pattern Recognition, Second Edition”, Academic Press, Elsevier, 2003.

[29] [En ligne]. Available: https://interstices.info/jcms/p_88807/marvin-minsky-un-des-cerveaux-de-l-intelligenceartificielle .

[30] V. Bjorn, “One Finger at a Time: Best Practices for Biometric Security,” Banking Information Source (Document ID: 1697301411), April, 2009.

[31] Chung et al, K.W. Cheung, J.T. Kwok, M.H. Law, K.C. Tsui,"Mining customer product ratings for personalized marketing", Decision Support Systems, vol. 35, issue 2, pp. 231-243, 2003.

[32] Kaastra et Boyd, I. Kaastra, M. Boyd, "Designing a neural network for forecasting financial and economic time series", Neurocomputing. Vol. 10, no. 3, pp. 215-236. 1996.

[33] Gupta et al, S.K. Gupta, W.C. Regli, D.S. Nau, "Manufacturing Feature Instances: Which Ones to Recognize?", Symposium on Solid Modeling and Applications, pp. 141152, 1995.

[34] Hippert et al, H.S. Hippert, C.E. Pedreira, R.C. Souza, "Neural networks for shortterm load forecasting: a review and evaluation", IEEE Transactions on Power Systems, vol. 16, Part 1, pp. 44-55, 2001.

[35] Munich et Perona, M.E. Munich, P. Perona, "Visual Input for Pen-Based Computers", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, issue 3, 2002.

[36] Yang et al, M.H. Yang, D.J. Kriegman, N. Ahuja, "Detecting Faces in Images: A 51 Survey", IEEE Transactions On Pattern Analysis And Machine Intelligence PAMI, vol.24; part 1, pp. 34-58, 2002.

[37] Theodoridis et Koutroumbas, S. Theodoridis, K. Koutroumbas, “Pattern Recognition, Second Edition”, Academic Press, Elsevier, 2003.

[38] [En ligne]. Available: https://www.interstices.info/jcms/c_5952/histoire-du-traitement-d-images.

[39] ‘’Mise au Point d’une Application de Reconnaissance de Formes’’, Mémoire de fin d’études pour l’obtention du diplôme de Master en Informatique, DJABEUR DJEZZAR Mohammed Rafik, BENKADA Feth-a.

[40] Kullback, S. 1959, Information Theory and Statistics, Wiley, New York.

[41] Satish S. Hiremath and K.V. Suresh. CHARACTER RECOGNITION OF VIDEO SUBTITLES. Department of Electronic and Communication Engineering, Siddaganga Institute of Technology, India,November 2016.

[42] C. Dorai, H. Aradhye, and J.-C. Shim. End-to-end video text recognition for multimedia content analysis. In ICME, pages 601–604, 2001.

[43] Khaoula Elagouni, Christophe Garcia, Pascale Sébillot. Reconnaissance automatique de texte dans des vidéos à l’aide d’un OCR et de connaissances linguistiques. GRETSI, Sep 2011, Bordeaux, France.

[44] S. Sabour, N. Frosst et G. E. Hinton, «Dynamic Routing Between Capsules,» (2017-10-26).

[45] Q. C. ,. C. W. ,. Z. L. Wangsheng Zhu, « A segmentation algorithm based on imageprojection for complex text layout,» October 2017 .

[46] A. A. e. S. Amidi, «CS 230 – Apprentissage profond Petites astuces,» Université de stanford, 6 janvier 2019.

[47] M. Pechyonkin, «Understanding Hinton’s Capsule Networks. Part 2. How Capsules Work.,» 15 Nov 2017. [En ligne]. Available: https://pechyonkin.me/capsules-2/.

[48] M. Pechyonkin, «Understanding Hinton’s Capsule Networks. Part 3. Dynamic Routing Between Capsules,» 29 Nov 2017. [En ligne]. Available: https://pechyonkin.me/capsules-3/.

[49] A. A. e. S. Amidi, «CS 230 – Apprentissage profond : Réseaux de neurones Convolutionnels,»

Université de stanford, 6 janvier 2019.

[50] « Classification des images avec les réseaux de neurones Convolutionnels », Mémoire de fin d'études pour l'obtention du diplôme de Master en informatique, M. Mokri Mohammed Zakaria.

[51] [En ligne]. Available: https://pypi.org/project/Pillow/.

[52] [En ligne]. Available: https://pypi.org/project/tensorflow-gpu/.

[53] [En ligne]. Available: https://fr.wikipedia.org/wiki/Keras.

[54] [En ligne]. Available: https://fr.wikipedia.org/wiki/Tkinter.

[55] [En ligne]. Available: https://www.jetbrains.com/fr-fr/pycharm/.

[56] [En ligne]. Available: https://fr.wikipedia.org/wiki/OpenCV.

[57] [En ligne]. Available: https://docs.python.org/fr/3/tutorial/.

[58] Karatzas, D.; Gomez-Bigorda, L.; Nicolaou, A.; Ghosh, S.; Bagdanovl, A.; Iwamura, M.; Matas, J.; Neumann, L.; Chandrasekhar, V.R.; Lu, S.; et al. ICDAR2015 competition on robust reading. In Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR), Oradea, Romania, 11–12 June 2015; pp. 1156–1160.

[59] [En ligne]. Available : https://fr.wikipedia.org/wiki/Vision_par_ordinateur.

[60] [En ligne]. Available : https://fr.wikipedia.org/wiki/Matplotlib.

[61] [En ligne]. Available : https://fr.wikipedia.org/wiki/JQuery.

[62] [En ligne]. Available : https://fr.wikipedia.org/wiki/Flask_(framework).

[63] [En ligne]. Available : https://fr.wikipedia.org/wiki/NumPy.